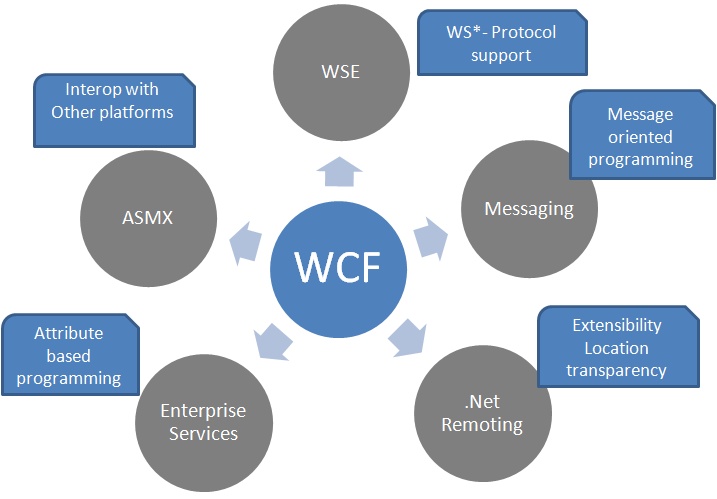
# Introduction to WCF

Windows Communication Foundation (Code named Indigo) is a programming platform and runtime system for building, configuring and deploying network-distributed services. It is the latest service oriented technology; Interoperability is the fundamental characteristics of WCF. It is unified programming model provided in .Net Framework 3.0. WCF is a combined features of Web Service, Remoting, MSMQ and COM+. WCF provides a common platform for all .NET communication.

Below figures shows the different technology combined to form WCF.



### Advantage

1. WCF is interoperable with other services when compared to .Net Remoting,where the client and service have to be .Net.
2. WCF services provide better reliability and security in compared to ASMX web services.
3. In WCF, there is no need to make much change in code for implementing the security model and changing the binding. Small changes in the configuration will make your requirements.
4. WCF has integrated logging mechanism, changing the configuration file settings will provide this functionality. In other technology developer has to write the code.

### Disadvantage

Making right design for your requirement is little bit difficult. I will try to help you on solving these difficulties in the following article.

# Development Tools

WCF application can be developed by the Microsoft Visual Studio. Visual studio is available at different edition. You can use Visual Studio 2008 Expression edition for the development.

<http://www.microsoft.com/express/product/default.aspx>

Visual Studio 2008 SDK 1.1

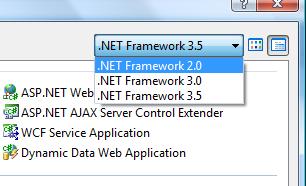
[http://www.microsoft.com/downloads/details.aspx?FamilyID=59ec6ec3-4273-48a3-ba25-dc925a45584d...](http://www.microsoft.com/downloads/details.aspx?FamilyID=59ec6ec3-4273-48a3-ba25-dc925a45584d&displaylang=en)

### Microsoft Visual Studio 2008

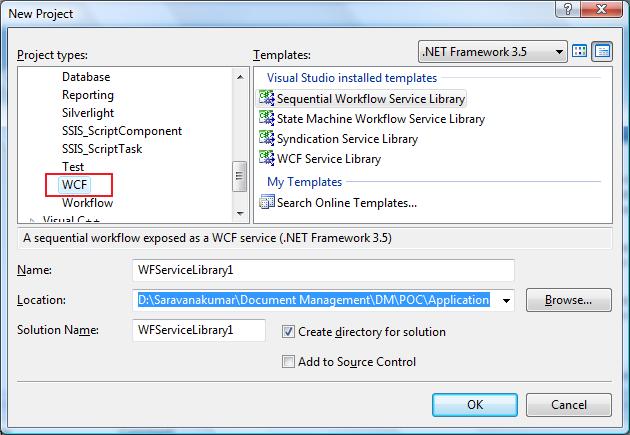
Microsoft Visual studio 2008 provides new features for WCF compared to Visual Studio 2005. These are the new features added to VS 2008.

#### Multi-targeting

You can create application in different framework like Framework 2.0, 3.0 and 3.5

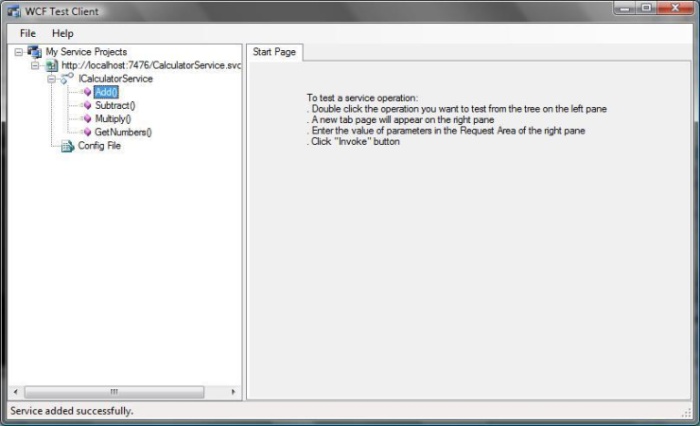


#### Default template is available for WCF



#### WCF - Test Client tools for testing the WCF service.

Microsoft provides inbuilt application to test the WCF application. This can be done by opening the Visual Studio command prompt and type the *wcfClient Serviceurl* shows below. This will help the developer to test the service before creating the client application.



1. **WCF services can be debugged now in Visual Studio 2008. Wcfsvchost.exe will do it for you because service will be self hosted when you start debugging.**

**Difference between WCF and Web service**

Web service is a part of WCF. WCF offers much more flexibility and portability to develop a service when comparing to web service. Still we are having more advantages over Web service, following table provides detailed difference between them.

|  |  |  |
| --- | --- | --- |
| **Features** | **Web Service** | **WCF** |
| Hosting | It can be hosted in IIS | It can be hosted in IIS, windows activation service, Self-hosting, Windows service |
| Programming | [WebService] attribute has to be added to the class | [ServiceContraact] attribute has to be added to the class |
| Model | [WebMethod] attribute represents the method exposed to client | [OperationContract] attribute represents the method exposed to client |
| Operation | One-way, Request- Response are the different operations supported in web service | One-Way, Request-Response, Duplex are different type of operations supported in WCF |
| XML | System.Xml.serialization name space is used for serialization | System.Runtime.Serialization namespace is used for serialization |
| Encoding | XML 1.0, MTOM(Message Transmission Optimization Mechanism), DIME, Custom | XML 1.0, MTOM, Binary, Custom |
| Transports | Can be accessed through HTTP, TCP, Custom | Can be accessed through HTTP, TCP, Named pipes, MSMQ,P2P, Custom |
| Protocols | Security | Security, Reliable messaging, Transactions |

# EndPoint

WCF Service is a program that exposes a collection of Endpoints. Each Endpoint is a portal for communicating with the world.

All the WCF communications are take place through end point. End point consists of three components.

### Address

Basically URL, specifies where this WCF service is hosted .Client will use this url to connect to the service. e.g

http://localhost:8090/MyService/SimpleCalculator.svc

### Binding

Binding will describes how client will communicate with service. There are different protocols available for the WCF to communicate to the Client. You can mention the protocol type based on your requirements.

A binding has several characteristics, including the following:

* Transport -Defines the base protocol to be used like HTTP, Named Pipes, TCP, and MSMQ are some type of protocols.
* Encoding (Optional) - Three types of encoding are available-Text, Binary, or Message Transmission Optimization Mechanism (MTOM). MTOM is an interoperable message format that allows the effective transmission of attachments or large messages (greater than 64K).
* Protocol(Optional) - Defines information to be used in the binding such as Security, transaction or reliable messaging capability

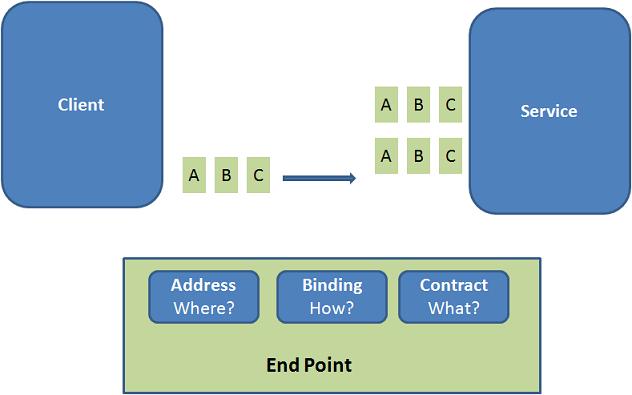
The following table gives some list of protocols supported by WCF binding.

|  |  |
| --- | --- |
| **Binding** | **Description** |
| BasicHttpBinding | Basic Web service communication. No security by default |
| WSHttpBinding | Web services with WS-\* support. Supports transactions |
| WSDualHttpBinding | Web services with duplex contract and transaction support |
| WSFederationHttpBinding | Web services with federated security. Supports transactions |
| MsmqIntegrationBinding | Communication directly with MSMQ applications. Supports transactions |
| NetMsmqBinding | Communication between WCF applications by using queuing. Supports transactions |
| NetNamedPipeBinding | Communication between WCF applications on same computer. Supports duplex contracts and transactions |
| NetPeerTcpBinding | Communication between computers across peer-to-peer services. Supports duplex contracts |
| NetTcpBinding | Communication between WCF applications across computers. Supports duplex contracts and transactions |

### Contract

Collection of operation that specifies what the endpoint will communicate with outside world. Usually name of the Interface will be mentioned in the Contract, so the client application will be aware of the operations which are exposed to the client. Each operation is a simple exchange pattern such as one-way, duplex and request/reply.

Below figure illustrate the functions of Endpoint



#### Example:

Endpoints will be mentioned in the web.config file on the created service.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint

address="http://localhost:8090/MyService/MathService.svc" contract="IMathService"

binding="wsHttpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

\

# EndPoint

WCF Service is a program that exposes a collection of Endpoints. Each Endpoint is a portal for communicating with the world.

All the WCF communications are take place through end point. End point consists of three components.

### Address

Basically URL, specifies where this WCF service is hosted .Client will use this url to connect to the service. e.g

http://localhost:8090/MyService/SimpleCalculator.svc

### Binding

Binding will describes how client will communicate with service. There are different protocols available for the WCF to communicate to the Client. You can mention the protocol type based on your requirements.

A binding has several characteristics, including the following:

* Transport -Defines the base protocol to be used like HTTP, Named Pipes, TCP, and MSMQ are some type of protocols.
* Encoding (Optional) - Three types of encoding are available-Text, Binary, or Message Transmission Optimization Mechanism (MTOM). MTOM is an interoperable message format that allows the effective transmission of attachments or large messages (greater than 64K).
* Protocol(Optional) - Defines information to be used in the binding such as Security, transaction or reliable messaging capability

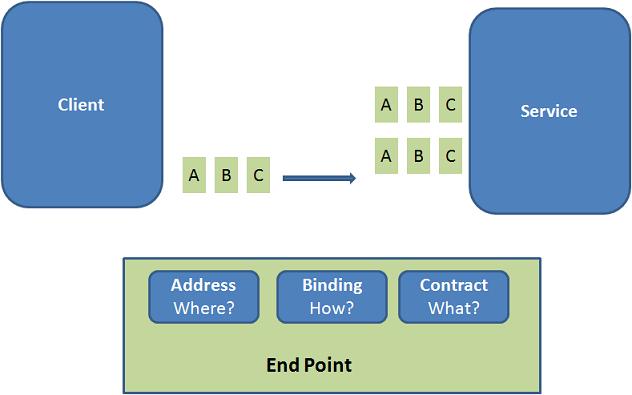
The following table gives some list of protocols supported by WCF binding.

|  |  |
| --- | --- |
| **Binding** | **Description** |
| BasicHttpBinding | Basic Web service communication. No security by default |
| WSHttpBinding | Web services with WS-\* support. Supports transactions |
| WSDualHttpBinding | Web services with duplex contract and transaction support |
| WSFederationHttpBinding | Web services with federated security. Supports transactions |
| MsmqIntegrationBinding | Communication directly with MSMQ applications. Supports transactions |
| NetMsmqBinding | Communication between WCF applications by using queuing. Supports transactions |
| NetNamedPipeBinding | Communication between WCF applications on same computer. Supports duplex contracts and transactions |
| NetPeerTcpBinding | Communication between computers across peer-to-peer services. Supports duplex contracts |
| NetTcpBinding | Communication between WCF applications across computers. Supports duplex contracts and transactions |

### Contract

Collection of operation that specifies what the endpoint will communicate with outside world. Usually name of the Interface will be mentioned in the Contract, so the client application will be aware of the operations which are exposed to the client. Each operation is a simple exchange pattern such as one-way, duplex and request/reply.

Below figure illustrate the functions of Endpoint



#### Example:

Endpoints will be mentioned in the web.config file on the created service.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint

address="http://localhost:8090/MyService/MathService.svc" contract="IMathService"

binding="wsHttpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

# Contracts and Service Host

## Contracts

In WCF, all services are exposed as contracts. Contract is a platform-neutral and standard way of describing what the service does. Mainly there are four types of contracts available in WCF

### Service Contract

Service contracts describe the operation that service can provide. For Eg, a Service provide to know the temperature of the city based on the zip code, this service is called as Service contract. It will be created using Service and Operational Contract attribute.

To know more on Service contract see [Service contract](http://wcftutorial.net/Service-Contract.aspx) tutorial.

### Data Contract

Data contract describes the custom data type which is exposed to the client. This defines the data types, that are passed to and from service. Data types like int, string are identified by the client because it is already mention in XML schema definition language document, but custom created class or data types cannot be identified by the client e.g. Employee data type. By using DataContract we can make client to be aware of Employee data type that are returning or passing parameter to the method.

To know more on DataContract see [DataContract](http://wcftutorial.net/Data-Contract.aspx) tutorial.

### Message Contract

Default SOAP message format is provided by the WCF runtime for communication between Client and service. If it is not meeting your requirements then we can create our own message format. This can be achieved by using Message Contract attribute.

To know more on Message Contract see [Message contract](http://wcftutorial.net/Message-Contract.aspx) tutorial.

### Fault Contract

Suppose the service I consumed is not working in the client application. I want to know the real cause of the problem. How I can know the error? For this we are having Fault Contract. Fault Contract provides documented view for error occurred in the service to client. This helps us to easy identity, what error has occurred.

To know more on Fault Contract see [Fault Contract](http://wcftutorial.net/Fault-Contract.aspx) tutorial.

# Service Host

Service Host object is in the process of hosting the WCF service and registering endpoints. It loads the service configuration endpoints, apply the settings and start the listeners to handle the incoming request. *System.ServiceModel.ServiceHost* namespace hold this object. This object is created while self hosting the WCF service.

In the below example you can find that WCF service is self hosted using console application.

//Creating uri for the hosting the service

Uri uri = new Uri("http://localhost/CategoryService");

//Creating the host object for MathService

ServiceHost host = new ServiceHost(typeof(CategoryService), uri);

//Adding endpoint to the Host object

host.AddServiceEndpoint(typeof(ICategoryService),new WSHttpBinding(), uri);

host.Open(); //Hosting the Service

Console.WriteLine("Waiting for client invocations");

Console.ReadLine();

host.Close();

# Message and Channel

### Message

WCF Message is the unit of data exchange between client and service. It consists of several parts, including a body and headers.

### WCF Runtime

WCF runtime is the set of object responsible for sending and receiving message. For example formatting the message, applying security and transmitting and receiving message using various protocol.

### Channels:

Channels are the core abstraction for sending message to and receiving message from an Endpoint. Broadly we can categories channels as

#### Transport Channels

**- Handles sending and receiving message from network. Protocols like HTTP, TCP name pipes and MSMQ.**

#### Protocol Channels

- Implements SOAP based protocol by processing and possibly modifying message. e.g. WS-Security and WS-Reliability.

# WCF Client and Metadata

### WCF Client

WCF client is a client application creates to expose the service operations as method. Any application can host a WCF client, including an application that host a service. Therefore it is possible to create a service that includes WCF clients of other services.

A client application is a managed application that uses a WCF client to communicate with another application. To create a client application for a WCF service requires the following steps:

1. Get the Proxy class and service end point information

Using *SvcUtil.exe* we can create proxy class for the service and configuration information for endpoints. Example type the following sentence in the Visual studio command prompt, this will generate the class file and configuration file which contain information about the endpoints.

*svcutil /language:vb /out:ClientCode.vb /config:app.config* *http://localhost:8090/MyService/SimpleCalculator.svc?wsdl*

1. Call operations.

Add this class files in the client application. Then create the object for this class and invoke the service operation. Configuration information we got from the above step has to be added to the client application configuration file. When the client application calls the first operation, WCF automatically opens the underlying channel. This underlying channel is closed, when the object is recycled.

//Creating the proxy on client side

MyCalculatorServiceProxy.MyServiceProxy proxy

= new MyCalculatorServiceProxy.MyServiceProxy();

Console.WriteLine("Counter: " + proxy.MyMethod());

1. Close the WCF client object.

After using the object created in the above steps, we have to dispose the object. Channel will be closed with the service, when the object is cleared.

### Metadata

Characteristics of the service are described by the metadata. This metadata can be exposed to the client to understand the communication with service. Metadata can be set in the service by enabling the ServiceMetadata node inside the servcieBehaviour node of the service configuration file.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint address="" contract="IMathService"

binding="wsHttpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

# WCF Architecture

The following figure illustrates the major components of WCF.

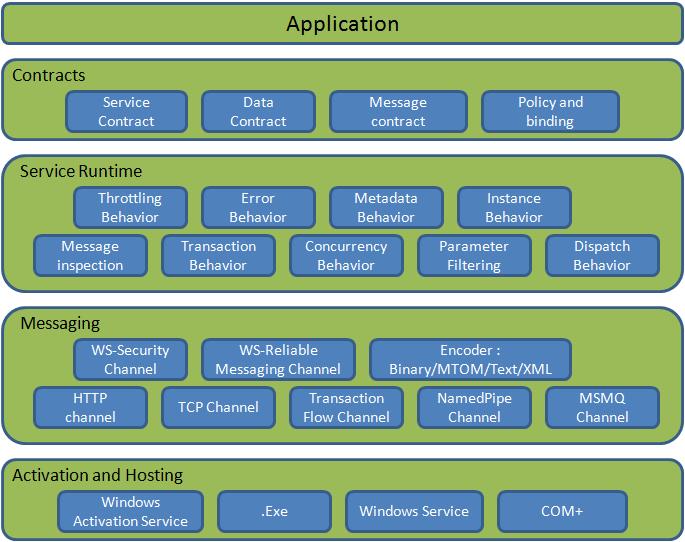


Figure 1: WCF Architecture

### Contracts

Contracts layer are next to that of Application layer. Developer will directly use this contract to develop the service. We are also going to do the same now. Let us see briefly what these contracts will do for us and we will also know that WCF is working on message system.

### Service contracts

- Describe about the operation that service can provide. Example, Service provided to know the temperature of the city based on the zip code, this service we call as Service contract. It will be created using Service and Operational Contract attribute.

### Data contract

- It describes the custom data type which is exposed to the client. This defines the data types, are passed to and from service. Data types like int, string are identified by the client because it is already mention in XML schema definition language document, but custom created class or datatype cannot be identified by the client e.g. Employee data type. By using DataContract we can make client aware that we are using Employee data type for returning or passing parameter to the method.

### Message Contract

- Default SOAP message format is provided by the WCF runtime for communication between Client and service. If it is not meeting your requirements then we can create our own message format. This can be achieved by using Message Contract attribute.

### Policies and Binding

- Specify conditions required to communicate with a service e.g security requirement to communicate with service, protocol and encoding used for binding.

### Service Runtime

- It contains the behaviors that occur during runtime of service.

* Throttling Behavior- Controls how many messages are processed.
* Error Behavior - Specifies what occurs, when internal error occurs on the service.
* Metadata Behavior - Tells how and whether metadata is available to outside world.
* Instance Behavior - Specifies how many instance of the service has to be created while running.
* Transaction Behavior - Enables the rollback of transacted operations if a failure occurs.
* Dispatch Behavior - Controls how a message is processed by the WCF Infrastructure.

### Messaging

- Messaging layer is composed of channels. A channel is a component that processes a message in some way, for example, by authenticating a message. A set of channels is also known as a channel stack. Channels are the core abstraction for sending message to and receiving message from an Endpoint. Broadly we can categories channels as

* Transport Channels

Handles sending and receiving message from network. Protocols like HTTP, TCP, name pipes and MSMQ.

* Protocol Channels

Implements SOAP based protocol by processing and possibly modifying message. E.g. WS-Security and WS-Reliability.

### Activation and Hosting

- Services can be hosted or executed, so that it will be available to everyone accessing from the client. WCF service can be hosted by following mechanism

* IIS

Internet information Service provides number of advantages if a Service uses Http as protocol. It does not require Host code to activate the service, it automatically activates service code.

* Windows Activation Service

(WAS) is the new process activation mechanism that ships with IIS 7.0. In addition to HTTP based communication, WCF can also use WAS to provide message-based activation over other protocols, such as TCP and named pipes.

* Self-Hosting

WCF service can be self hosted as console application, Win Forms or WPF application with graphical UI.

* Windows Service

WCF can also be hosted as a Windows Service, so that it is under control of the Service Control Manager (SCM).

## Introduction

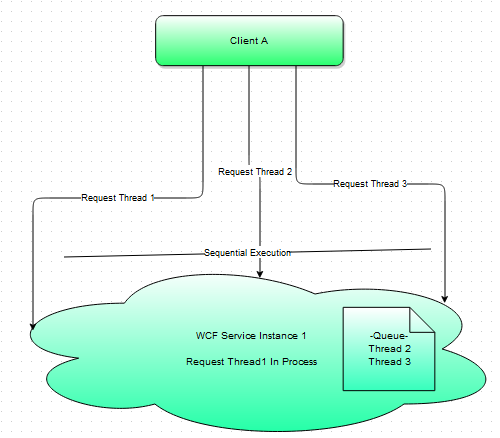
This article will explain how we can achieve through puts, reliability and performance using various WCF Service behavior techniques. Most application differs widely in their scalability and performance strategies. WCF service behaviors provide various possible ways to configure key run time behaviors. We will go through various service behavior elements one by one.

## Concurrency

By default WCF service handles only single request at a time and all other service request threads are queued and processed one by one. Concurrency element allows clients to send multiple requests at the same time, but service implementation should be break free of scenarios like dead locks.

There are three types of Concurrency modes.

1. **Single(Default)**: Concurrency mode single will allows only one at a time to the service instance. And all other pending requests will be maintained in queue and processed one by one. Whenever a new request comes dispatcher gets a lock before entering in to the code. Refer Figure 1.



***Figure 1.***

ConcurrencyMode Single is defined as follows:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

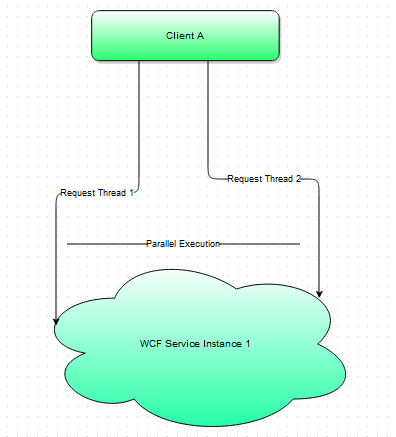
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single)]

public class VisitorCount : IVisitorCount

{

}

1. **Multiple**: This will allow parallel requests, and that are processed at the same time by spawning separate thread for each and every request. Refer Figure 2.



***Figure 2***

Here is how we would define ConcurrencyMode Multiple.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

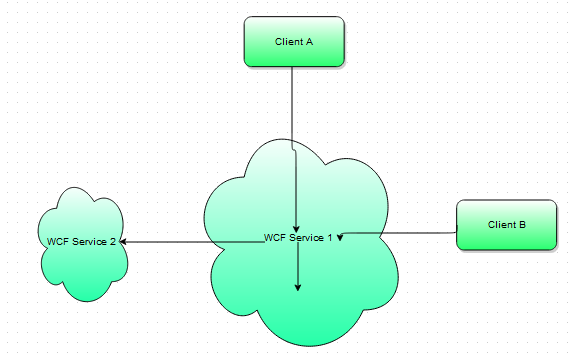
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple)]

public class VisitorCount : IVisitorCount

{

}

1. **Re-entrant**: Whenever a client calls WCF service a thread lock will be assigned to this client call. Let us consider a scenario where Service 1 makes an external call to Service 2. Thread lock won’t be released until entire service call is completed. So all other client requests are in waiting state. Re-entrant allows the client to release the lock before making call to the external service (Service 2) and this will allow other client to use the Service 1 facilities until Service 2 process is completed. Refer Figure 3.



***Figure 3***

Decorating service class with ConcurrencyMode Reentrant.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Reentrant)]

public class VisitorCount : IVisitorCount

{

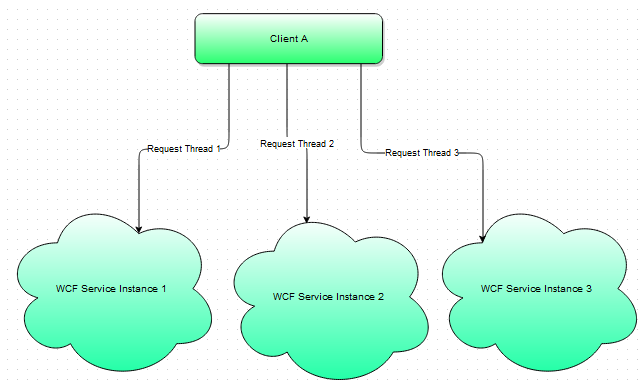
}

Concurrency mode can increase the amount of passing through the service methods and hence overall service performance increases.

## Instance Context Mode

WCF Instancing decides how objects are created and refers to the life time of the service object. Whenever clients make request runtime will create service objects to provide the response. Through instancing we ca control how long this service instance wants to be retained. For that we are using the three instancing modes.

1. **Per Call:**In this scenario, all the calls to the service become stateless. And for every thread request a new service instance will be created. This will work with all the service bindings. Refer Figure 4.



***Figure 4.***

Here is how we would define InstanceContext Mode Percall.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single,

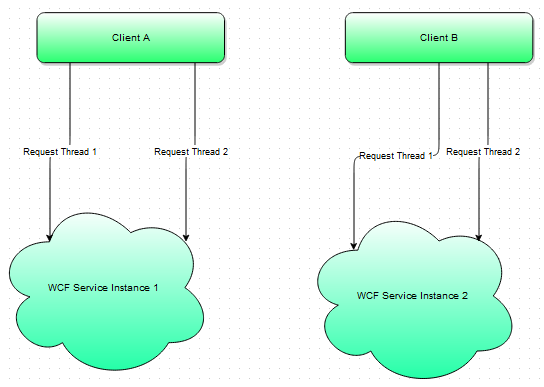
InstanceContextMode = InstanceContextMode.PerCall)]

public class VisitorCount : IVisitorCount

{

}

1. **Per Session**: The life time of service object is independent of the life time of Client channels, so  this will create a new service object whenever a new communication session is established and disposed after that. Each client channel gets a dedicated service instance and the subsequent calls in the same session are handled by the same Service object. This is the default value for Instancing Context. And this will work with all bindings except basicHttpBindings. Refer Figure 5.



***Figure 5.***

Here is how we would define InstanceContextMode PerSession.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

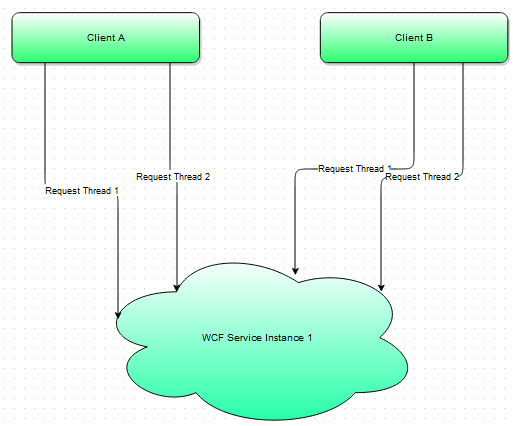
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single,InstanceContextMode = InstanceContextMode.PerSession)]

public class VisitorCount : IVisitorCount

{

}

1. **Single**: This will help us to share the data globally. We can create only one instance and the same instance will be reused on the subsequent calls. Same like Per Session this will work with all bindings except basicHttpBinding. Single ton instance will not be disposed until service host is down. Refer Figure 6.



***Figure 6***

Here is how we would define InstanceContext mode Single.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single,InstanceContextMode = InstanceContextMode.Single)]

public class VisitorCount : IVisitorCount

{

}

## Service Throttling

WCF allows you to throttle load on a particular service type.  
  
This includes

1. Maximum number of Concurrent sessions
2. Maximum number of Concurrent calls
3. Maximum number of Concurrent instances

Whenever these values exceeded, the callers will be added to the queue and processed FIFO order. We can configure throttling behavior in application configuration file as follows.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

<serviceBehaviors>

<behavior name="visitorCountServiceBehavior">

<serviceThrottling maxConcurrentCalls="5" maxConcurrentInstances="10" maxConcurrentSessions="10">

</serviceThrottling>

</behavior>

</serviceBehaviors>

## Using the code

Let us create a sample application to demonstrate different service behaviors. First create an interface and decorate it with  ServiceContract and OperationalContract attributes to specify that these operations of this interface can be used by the client applications.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

[ServiceContract]

public interface IVisitorCount

{

[OperationContract]

int GetVisitorCount();

}

Now we have to create a service class to implement above interface method. And we have to decorate service class with different service behaviors, following code explains Concurrency Mode Single and Instance context mode single.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single,InstanceContextMode = InstanceContextMode.Single)]

public class VisitorCount : IVisitorCount

{

int \_visitorCount = 0;

public VisitorCount()

{

Console.WriteLine("New Service Instance Created");

}

public int GetVisitorCount()

{

\_visitorCount++;

return \_visitorCount;

}

}

Now let us see how we can make this service available to client applications, Configuration file App.Config contents are given below.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

<configuration>

<system.web>

<compilation debug="true">

</compilation></system.web>

<system.servicemodel>

<services>

<service name="InstanceContextModeService.VisitorCount">

<endpoint binding="wsHttpBinding" contract="InstanceContextModeService.IVisitorCount">

<identity>

<dns value="localhost">

</dns></identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange">

<host>

<baseAddresses>

<add baseaddress="http://localhost:8732/InstanceContextModeService/VisitorCount/">

</baseAddresses>

</add></host>

</endpoint></service>

</services>

<behaviors>

<servicebehaviors>

<behavior>

<servicemetadata httpgetenabled="True">

<servicedebug includeexceptiondetailinfaults="False">

<servicethrottling maxconcurrentcalls="5" maxconcurrentinstances="10" maxconcurrentsessions="10">

</servicethrottling></servicedebug></servicemetadata><</behavior>

</servicebehaviors>

</behaviors>

</system.servicemodel>

</configuration>

Following service host code will host VisitorCounter service to the client applications.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

class Program

{

private static ServiceHost host = null;

static void Main(string[] args)

{

host = new ServiceHost(typeof(InstanceContextModeService.VisitorCount));

host.Opened += new EventHandler(host\_Opened);

host.Closed += new EventHandler(host\_Closed);

host.Open();

Console.ReadKey();

host.Close();

Console.ReadKey();

}

static void host\_Closed(object sender, EventArgs e)

{

Console.WriteLine("Service Closed");

}

static void host\_Opened(object sender, EventArgs e)

{

Console.WriteLine("Service Started");

}

}

We have our service is hosted and defined. Following code explains how clients utilize service and service behaviors.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/443576/Implementing-WCF-Service-Behaviors)

static class Program

{

static void Main(string[] args)

{

VisitorCountClient client = new VisitorCountClient();

Console.WriteLine("First Call-->" + client.GetVisitorCount());

Console.ReadKey();

Console.WriteLine("Second Call-->" + client.GetVisitorCount());

Console.ReadKey();

Console.WriteLine("Third Call-->" + client.GetVisitorCount());

Console.ReadKey();

Console.WriteLine("Forth Call-->" + client.GetVisitorCount());

Console.ReadKey();

}

}

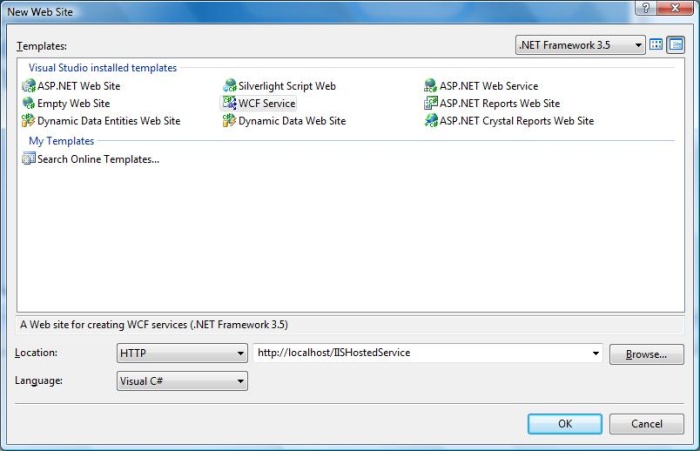
Download above attached sample application for your reference and go through different service behaviors by changing different behavior attributes as we discussed earlier

**IIS 5/6 Hosting**

The main advantage of hosting service in IIS is that, it will automatically launch the host process when it gets the first client request. It uses the features of IIS such as process recycling, idle shutdown, process health monitoring and message based activation. The main disadvantage of using IIS is that, it will support only HTTP protocol.

Let as do some hands on, to create service and host in IIS

**Step 1:**Start the Visual Studio 2008 and click File->New->Web Site. Select the 'WCF Service' and Location as http. This will directly host the service in IIS and click OK.



**Step 2:** I have created sample HelloWorld service, which will accept name as input and return with 'Hello' and name. Interface and implementation of the Service is shown below.

**IMyService.cs**

[ServiceContract]

public interface IMyService

{

[OperationContract]

string HelloWorld(string name);

}

**MyService.cs**

public class MyService : IMyService

{

#region IMyService Members

public string HelloWorld(string name)

{

return "Hello " + name;

}

#endregion

}

**Step 3:** Service file (.svc) contains name of the service and code behind file name. This file is used to know about the service.

**MyService.svc**

<%@ ServiceHost Language="C#" Debug="true"

Service="MyService" CodeBehind="~/App\_Code/MyService.cs" %>

**Step 4:** Server side configurations are mentioned in the config file. Here I have mention only one end point which is configured to 'wsHttpBinding', we can also have multiple end point with differnet binding. Since we are going to hosted in IIS. We have to use only http binding. We will come to know more on endpoints and its configuration in later tutorial. **Web.Config**

<system.serviceModel>

<services>

<service behaviorConfiguration="ServiceBehavior" name="MyService">

<endpoint address="http://localhost/IISHostedService/MyService.svc"

binding="wsHttpBinding" contract="IMyService">

<identity>

<dns value="localhost"/>

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="ServiceBehavior">

<!-- To avoid disclosing metadata information,

set the value below to false and remove the

metadata endpoint above before deployment -->

<serviceMetadata httpGetEnabled="true"/>

<!-- To receive exception details in faults for

debugging purposes, set the value below to true.

Set to false before deployment to avoid disclosing exception information -->

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

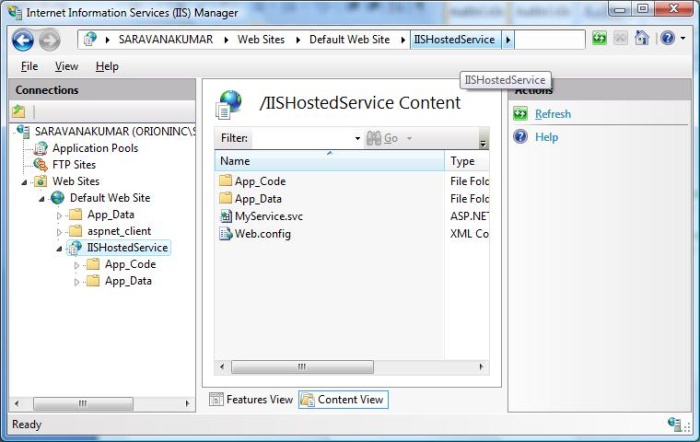
</serviceBehaviors>

</behaviors>

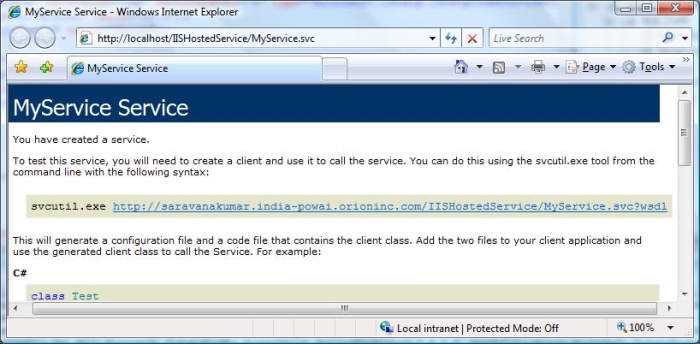
</system.serviceModel>

**Note:**

You need to mention the service file name, along with the Address mention in the config file. IIS Screen shot

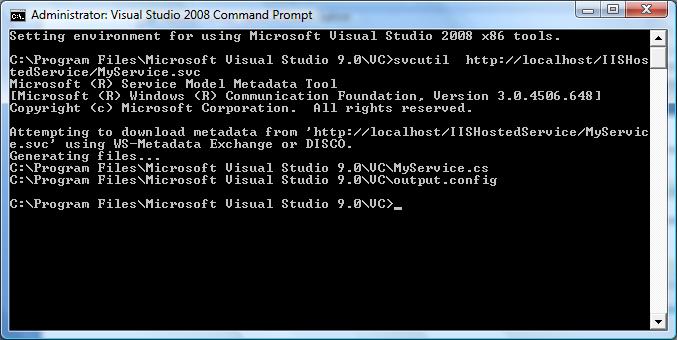


This screen will appear when we run the application.



**Step 5:** Now we successfully hosted the service in IIS. Next we have to consume this service in client application. Before creating the client application, we need to create the proxy for the service. This proxy is used by the client application, to interact with service. To create the proxy, run the Visual Studio 2008 command prompt. Using service utility we can create the proxy class and its configuration information.

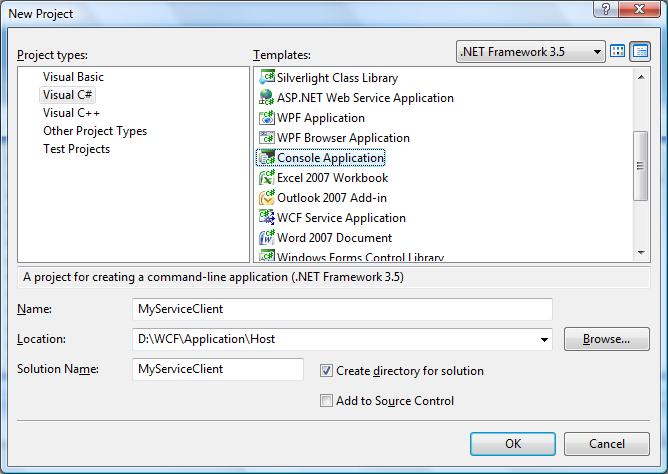
*svcutil http://localhost/IISHostedService/MyService.svc*



After executing this command we will find two file generated in the default location.

* MyService.cs - Proxy class for the WCF service
* output.config - Configuration information about the service.

**Step 6:** Now we will start creating the Console application using Visual Studio 2008(Client application).



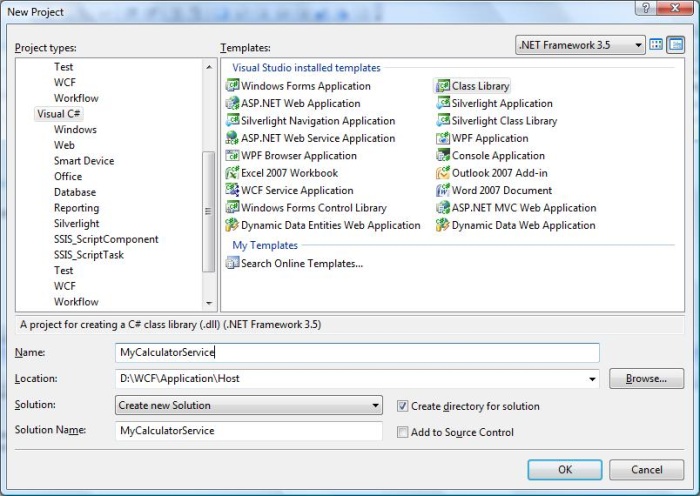
**Step 7:** Add the reference 'System.ServiceModel'; this is the core dll for WCF.

**Self Hosting**

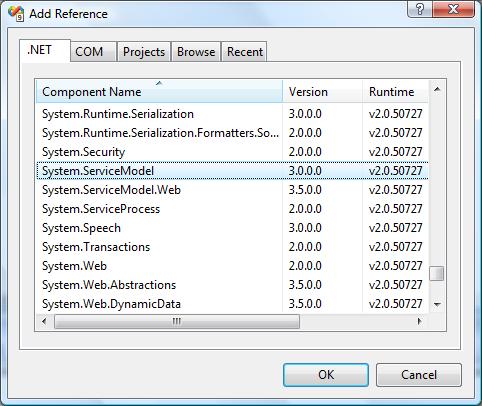
In web service, we can host the service only in IIS, but WCF provides the user to host the service in any application (e.g. console application, Windows form etc.). Very interestingly developer is responsible for providing and managing the life cycle of the host process. Service can also be in-pro i.e. client and service in the same process. Now let's us create the WCF service which is hosted in Console application. We will also look in to creating proxy using *'ClientBase'* abstract class.

**Note: Host process must be running before the client calls the service, which typically means you have to prelaunch it.**

**Step 1: First let's start create the Service contract and it implementation. Create a console application and name it as MyCalculatorService. This is simple service which return addition of two numbers.**

****

**Step 2: Add the System.ServiceModel reference to the project.**

****

**Step 3: Create an ISimpleCalculator interface, Add ServiceContract and OperationContract attribute to the class and function as shown below. You will know more information about these contracts in later session. These contracts will expose method to outside world for using this service.**

**IMyCalculatorService.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.ServiceModel;**

**namespace MyCalculatorService**

**{**

**[ServiceContract()]**

**public interface ISimpleCalculator**

**{**

**[OperationContract()]**

**int Add(int num1, int num2);**

**}**

**}**

**Step 4: MyCalculatorService is the implementation class for IMyCalculatorService interface as shown below.**

**MyCalculatorService.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace MyCalculatorService**

**{**

**class SimpleCalculator : ISimpleCalculator**

**{**

**public int Add(int num1, int num2)**

**{**

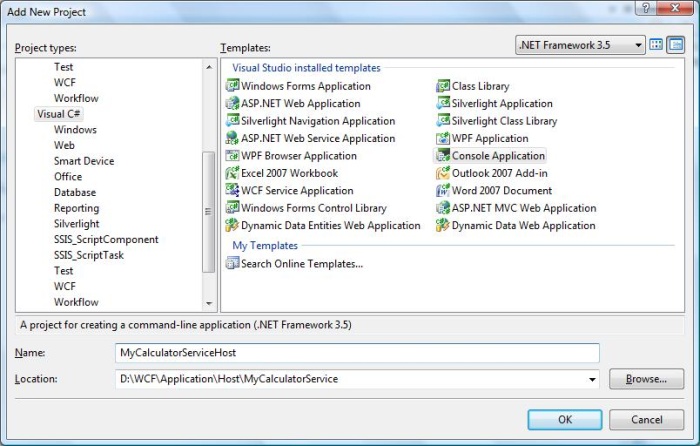
**return num1 + num2;**

**}**

**}**

**}**

**Step 5: Now we are ready with service. Let's go for implementing the hosting process. Create a new console application and name it as 'MyCalculatorServiceHost'**

****

**Step 6: *ServiceHost* is the core class use to host the WCF service. It will accept implemented contract class and base address as contractor parameter. You can register multiple base addresses separated by commas, but address should not use same transport schema.**

**Uri httpUrl**

**= new Uri("http://localhost:8090/MyService/SimpleCalculator");**

**Uri tcpUrl**

**= new Uri("net.tcp://localhost:8090/MyService/SimpleCalculator");**

**ServiceHost host**

**= new ServiceHost(typeof(MyCalculatorService.SimpleCalculator), httpUrl, tcpUrl);**

**Multiple end points can be added to the Service using *AddServiceEndpoint()* method. *Host.Open()* will run the service, so that it can be used by any client.**

**Step 7: Below code show the implementation of the host process.**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.ServiceModel;**

**using System.ServiceModel.Description;**

**namespace MyCalculatorServiceHost**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**//Create a URI to serve as the base address**

**Uri httpUrl = new Uri("http://localhost:8090/MyService/SimpleCalculator");**

**//Create ServiceHost**

**ServiceHost host**

**= new ServiceHost(typeof(MyCalculatorService.SimpleCalculator), httpUrl);**

**//Add a service endpoint**

**host.AddServiceEndpoint(typeof(MyCalculatorService.ISimpleCalculator)**

**, new WSHttpBinding(), "");**

**//Enable metadata exchange**

**ServiceMetadataBehavior smb = new ServiceMetadataBehavior();**

**smb.HttpGetEnabled = true;**

**host.Description.Behaviors.Add(smb);**

**//Start the Service**

**host.Open();**

**Console.WriteLine("Service is host at " + DateTime.Now.ToString());**

**Console.WriteLine("Host is running... Press <Enter> key to stop");**

**Console.ReadLine();**

**}**

**}**

**}**

**Step 8: Service is hosted, now we need to implement the proxy class for the client. There are different ways of creating the proxy**

* **Using SvcUtil.exe, we can create the proxy class and configuration file with end points.**
* **Adding Service reference to the client application.**
* **Implementing ClientBase<T> class**

**Of these three methods, Implementing ClientBase<T> is the best practice. If you are using rest two method, we need to create proxy class every time when we make changes in Service implementation. But this is not the case for ClientBase<T>. It will create the proxy only at runtime and so it will take care of everything.**

**MyCalculatorServiceProxy.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.ServiceModel;**

**using MyCalculatorService;**

**namespace MyCalculatorServiceProxy**

**{**

**public class MyCalculatorServiceProxy :**

**//WCF create proxy for ISimpleCalculator using ClientBase**

**ClientBase<ISimpleCalculator>,**

**ISimpleCalculator**

**{**

**public int Add(int num1, int num2)**

**{**

**//Call base to do funtion**

**return base.Channel.Add(num1, num2);**

**}**

**}**

**}**

**Step 9: In the client side, we can create the instance for the proxy class and call the method as shown below. Add proxy assembly as reference to the project.**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.ServiceModel;**

**namespace MyCalculatorServiceClient**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy ;**

**proxy= new MyCalculatorServiceProxy.MyCalculatorServiceProxy();**

**Console.WriteLine("Client is running at " + DateTime.Now.ToString());**

**Console.WriteLine("Sum of two numbers... 5+5 ="+proxy.Add(5,5));**

**Console.ReadLine();**

**}**

**}**

**}**

**Step 10 : End point (same as service) information should be added to the configuration file of the client application.**

**<?xml version="1.0" encoding="utf-8" ?>**

**<configuration>**

**<system.serviceModel>**

**<client>**

**<endpoint address ="http://localhost:8090/MyService/SimpleCalculator"**

**binding ="wsHttpBinding"**

**contract ="MyCalculatorService.ISimpleCalculator">**

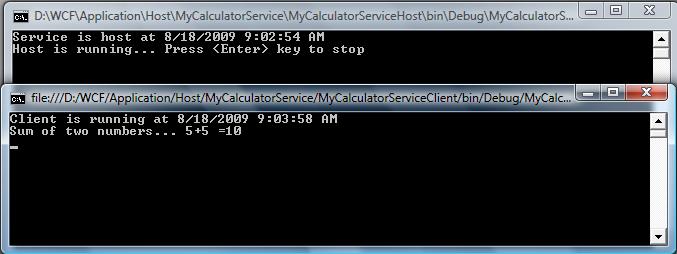
**</endpoint>**

**</client>**

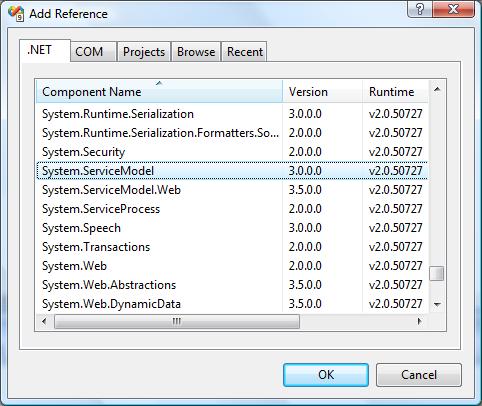
**</system.serviceModel>**

**</configuration>**

**Step 11: Before running the client application, you need to run the service. Output of the client application is shown below.**

****

**This self host shows advantage such as in-Pro hosting, programmatic access and it can be used when there need singleton service. I hope you have enjoyed the Self hosting session, now let go for hosting using Windows Activation service.**



**Step 8:** Create the object for the proxy class and call the HelloWorld method.

static void Main(string[] args)

{

//Creating Proxy for the MyService

MyServiceClient client = new MyServiceClient();

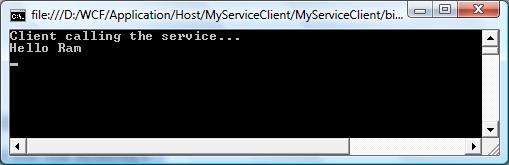
Console.WriteLine("Client calling the service...");

Console.WriteLine(client.HelloWorld("Ram"));

Console.Read();

}

**Step 9:** If we run the application we will find the output as shown below.



I hope you have enjoyed the Service hosted in IIS. Now let start the look on the self hosted service

# Windows Activation Service

Windows Activation service is a system service available with Windows vista and windows server 2008. It is available with IIS 7.0 and it is more powerful compared to IIS 6.0 because it supports Http, TCP and named pipes were IIS 6.0 supports only Http. It can be installed and configured separately.

Hosting WCF in Activation service takes many advantages such as process recycling, isolation, idle time management and common configuration system. WAS hosted service can be created using following steps

1. Enable WCF for non-http protocols
2. Create WAS hosted service
3. Enable different binding to the hosted service

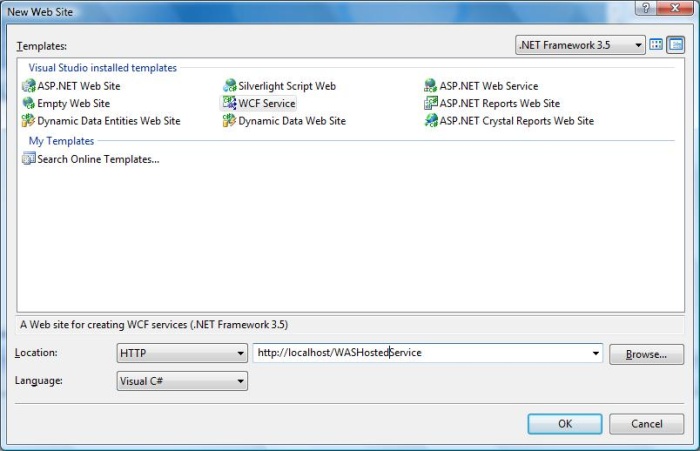
### Enable WCF for non-http protocols

Before Start creating the service we need to configure the system to support WAS. Following are the step to configure WAS.

1. Click Start -> Control Panel -> programs and Features and click 'Turn Windows Components On or Off' in left pane.
2. Expand 'Microsoft .Net Framework 3.0' and enable "Windows Communication Foundation HTTP Activation" and "Windows Communication Foundation Non- HTTP Activation".
3. Next we need to add Binding to the Default Web site. As an example, we will bind the default web site to the TCP protocol. Go to the Start menu -> Programs ->Accessories. Right click on the "Command Prompt" item, and select "Run as administrator" from the context menu.
4. Execute the following command
5. C:\Windows\system32\inetsrv> appcmd.exe set site "Default Web Site" -+bindings.[protocol='net.tcp', bindingInformation='808:\*']
6. That command adds the net.tcp site binding to the default web site by modifying the applicationHost.config file located in the "*C:\Windows\system32\inetsrv\config*" directory. Similarly we can add different protocols to the Default Web site.

### Create WAS hosted service

**Step 1:** Next we are going to create the service, Open the Visual Studio 2008 and click New->WebSite and select WCF Service from the template and Location as HTTP as shown below.



**Step 2:** Create the Contract by creating interface IMathService and add ServiceContract attribute to the interface and add OperationContract attribute to the method declaration.

**IMathService.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

[ServiceContract]

public interface IMathService

{

[OperationContract]

int Add(int num1, int num2);

[OperationContract]

int Subtract(int num1, int num2);

}

**Step 3:** Implementation of the IMathService interface is shown below.

**MathService.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

public class MathService : IMathService

{

public int Add(int num1, int num2)

{

return num1 + num2;

}

public int Subtract(int num1, int num2)

{

return num1 - num2;

}

}

**Step 4:** Service file is shown below.

**MathService.svc**

<%@ ServiceHost Language="C#" Debug="true" Service="MathService"

CodeBehind="~/App\_Code/MathService.cs" %>

**Step 5:** In web.Config file, create end point with 'netTcpBinding' binding and service metadata will be published using Metadata Exchange point. So create the Metada Exchange end point with address as 'mex' and binding as 'mexTcpBinding'. Without publishing the service Metadata we cannot create the proxy using net.tcp address (e.g svcutil.exe net.tcp://localhost/WASHostedService/MathService.svc )

**Web.Config**

<system.serviceModel>

<services>

<service name="MathService" behaviorConfiguration="ServiceBehavior">

<!-- Service Endpoints -->

<endpoint binding="netTcpBinding"

contract="IMathService" >

</endpoint>

<endpoint address="mex"

binding="mexTcpBinding" contract="IMetadataExchange"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="ServiceBehavior">

<!-- To avoid disclosing metadata information, set the value below

to false and remove the metadata endpoint above before deployment -->

<serviceMetadata httpGetEnabled="true"/>

<!-- To receive exception details in

faults for debugging purposes, set the value below to true.

Set to false before deployment to avoid disclosing

exception information -->

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

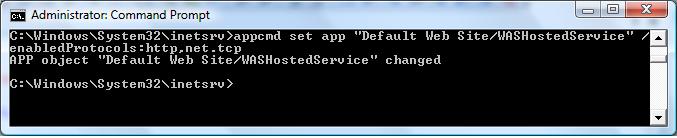
</serviceBehaviors></behaviors>

</system.serviceModel>

### Enable different binding to the hosted service

1. Go to the Start menu -> Programs ->Accessories. Right click on the "Command Prompt" item, and select "Run as administrator" from the context menu.
2. Execute the following command *C:\Windows\system32\inetsrv>appcmd set app "Default Web Site/WASHostedServcie" /enabledProtocols:http,net.tcp*

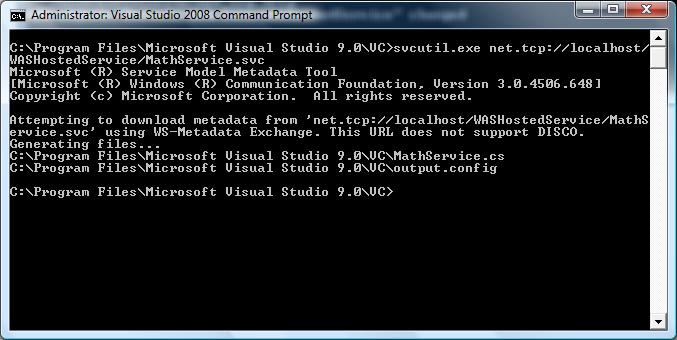
Output will be shown below.



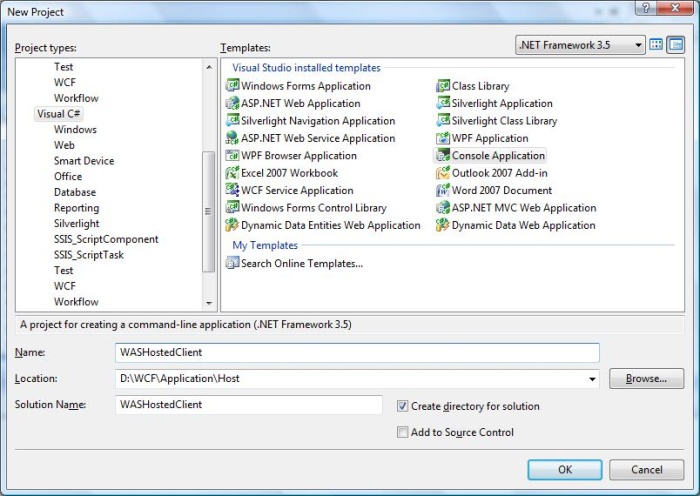
**Step 6:** Now the service is ready to use. Next we can create the proxy class using service uttility and add the proxy class to the client application. Creat the proxy class using Visual Studio Command prompt and execute the command

*svcutil.exe net.tcp://localhost/WASHostedService/MathService.svc*

Proxy and configuration file are generated in the corresponding location.



**Step 6:** Create the client application as shown below and add the reference '*System.ServiceModel*', this is the core dll for WCF.



**Step 8:** Add the proxy class and configuration file to the client application. Create the object for the MathServiceClient and call the method.

**Program.cs**

class Program

{

static void Main(string[] args)

{

MathServiceClient client = new MathServiceClient();

Console.WriteLine("Sum of two number 5,6");

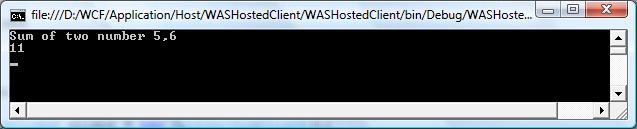
Console.WriteLine(client.Add(5,6));

Console.ReadLine();

}

}

The output will be shown as below.



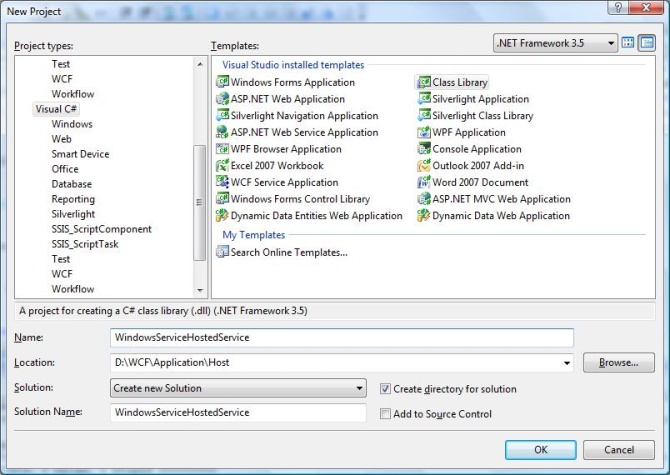
So this tutorial clearly explains about the hosting the WCF in Windows Activation Service. So next we can see how to host the

Bottom of Form

In this tutorial we are going to see the hosting WCF service in Windows service. We will use same set of code used for hosting the WCF service in Console application to this. This is same as hosting the service in IIS without message activated. There is some advantage of hosting service in Windows service.

* The service will be hosted, when system starts
* Process life time of the service can be controlled by Service Control Manager for windows service
* All versions of Windows will support hosting WCF service.

**Step 1:** Now let start create the WCF service, Open the Visual Studio 2008 and click New->Project and select Class Library from the template.



**Step 2:** Add reference *System.ServiceModel* to the project. This is the core assembly used for creating the WCF service.

**Step 3:** Next we can create the *ISimpleCalulator* interface as shown below. Add the Service and Operation Contract attribute as shown below.

**ISimpleCalculator.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ServiceModel;

namespace WindowsServiceHostedContract

{

[ServiceContract]

public interface ISimpleCalculator

{

[OperationContract]

int Add(int num1, int num2);

[OperationContract]

int Subtract(int num1, int num2);

[OperationContract]

int Multiply(int num1,int num2);

[OperationContract]

double Divide(int num1, int num2);

}

}

**Step 4:** Implement the *ISimpleCalculator* interface as shown below.

**SimpleCalulator.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace WindowsServiceHostedService

{

class SimpleCalculator

: ISimpleCalculator

{

public int Add(int num1, int num2)

{

return num1+num2;

}

public int Subtract(int num1, int num2)

{

return num1-num2;

}

public int Multiply(int num1, int num2)

{

return num1\*num2;

}

public double Divide(int num1, int num2)

{

if (num2 != 0)

return num1 / num2;

else

return 0;

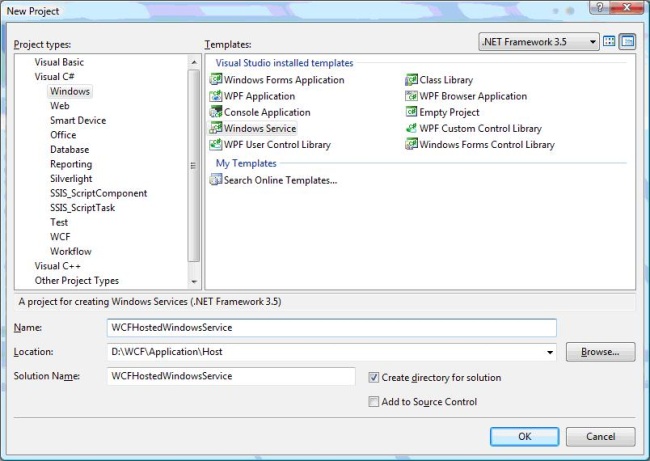
}

}

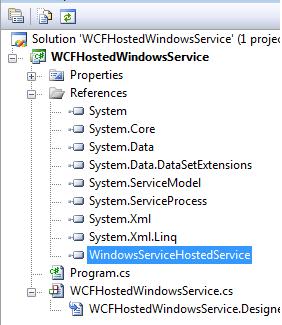
}

**Step 5:** Build the Project and get the dll. Now we are ready with WCF service, now we are going to see how to host the WCF Service in Windows service. **Note:** In this project, I have mention that we are creating both Contract and Service(implementation) are in same project. It is always good practice if you have both in different project.

**Step 6:** Open Visual Studio 2008 and Click New->Project and select Windows Service.



**Step 7:** Add the '*WindowsServiceHostedService.dll*' as reference to the project. This assembly will going to act as service.



**Step 8:** OnStart method of the service, we can write the hosting code for WCF. We have to make sure that we are using only one service host object. On stop method you need to close the Service Host. Following code show how to host WCF service in Windows service.

**WCFHostedWindowsService.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Linq;

using System.ServiceProcess;

using System.Text;

using System.ServiceModel;

using System.ServiceModel.Description;

namespace WCFHostedWindowsService

{

partial class WCFHostedWindowsService : ServiceBase

{

ServiceHost m\_Host;

public WCFHostedWindowsService()

{

InitializeComponent();

}

protected override void OnStart(string[] args)

{

if (m\_Host != null)

{

m\_Host.Close();

}

//Create a URI to serve as the base address

Uri httpUrl = new Uri("http://localhost:8090/MyService/SimpleCalculator");

//Create ServiceHost

m\_Host = new ServiceHost

(typeof(WindowsServiceHostedService.SimpleCalculator), httpUrl);

//Add a service endpoint

m\_Host.AddServiceEndpoint

(typeof(WindowsServiceHostedService.ISimpleCalculator), new WSHttpBinding(), "");

//Enable metadata exchange

ServiceMetadataBehavior smb = new ServiceMetadataBehavior();

smb.HttpGetEnabled = true;

m\_Host.Description.Behaviors.Add(smb);

//Start the Service

m\_Host.Open();

}

protected override void OnStop()

{

if (m\_Host != null)

{

m\_Host.Close();

m\_Host = null;

}

}

static void Main()

{

ServiceBase[] ServicesToRun;

ServicesToRun = new ServiceBase[]

{

new WCFHostedWindowsService()

};

ServiceBase.Run(ServicesToRun);

}

}

}

**Step 9:** In order to install the service we need to have the Installer class for the Windows service. So add new Installer class to the project, which is inherited from the *Installer* class. Please find the below code for mentioning the Service name, StartUp type etc of the service.

**ServiceInstaller.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.ServiceProcess;

using System.Configuration.Install;

using System.ComponentModel;

using System.Configuration;

namespace WCFHostedWindowsService

{

[RunInstaller(true)]

public class WinServiceInstaller : Installer

{

private ServiceProcessInstaller process;

private ServiceInstaller service;

public WinServiceInstaller()

{

process = new ServiceProcessInstaller();

process.Account = ServiceAccount.NetworkService;

service = new ServiceInstaller();

service.ServiceName = "WCFHostedWindowsService";

service.DisplayName = "WCFHostedWindowsService";

service.Description = "WCF Service Hosted";

service.StartType = ServiceStartMode.Automatic;

Installers.Add(process);

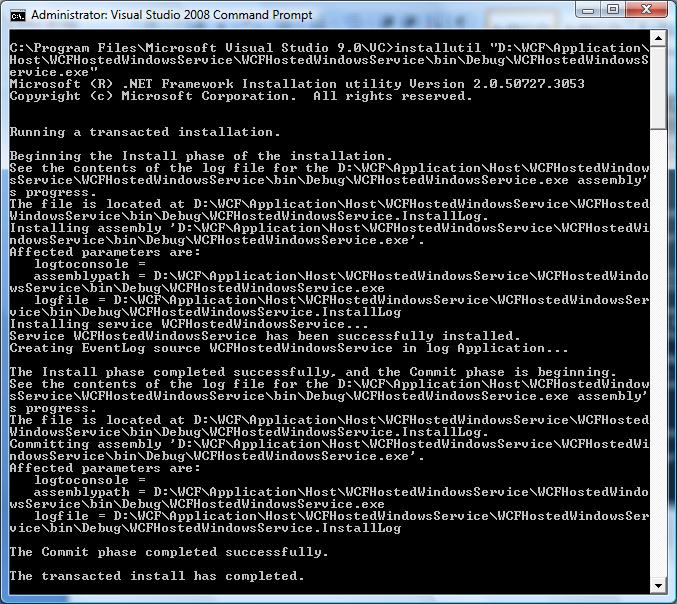
Installers.Add(service);

}

}

}

**Step 10:** Build the project, we will get the *WCFHostedWindowsService.exe*. Next we need to install the service using Visual Studio Command Prompt. So open the command prompt by clicking Start->All Programs-> Microsoft Visual Studio 2008-> Visual Studio Tools-> Visual Studio Command Prompt Using *installutil* utility application, you can install the service as shown below.



**Step 11:** Now service is Hosted sucessfully and we can create the proxy class for the service and start using in the client applcaiton.

# Types of Binding

Let us see more detailed on predefined binding

### BasicHttpBinding

* It is suitable for communicating with ASP.NET Web services (ASMX)-based services that comfort with WS-Basic Profile conformant Web services.
* This binding uses HTTP as the transport and text/XML as the default message encoding.
* Security is disabled by default
* This binding does not support WS-\* functionalities like WS- Addressing, WS-Security, WS-ReliableMessaging
* It is fairly weak on interoperability.

### WSHttpBinding

* Defines a secure, reliable, interoperable binding suitable for non-duplex service contracts.
* It offers lot more functionality in the area of interoperability.
* It supports WS-\* functionality and distributed transactions with reliable and secure sessions using SOAP security.
* It uses HTTP and HTTPS transport for communication.
* Reliable sessions are disabled by default.

### WSDualHttpBinding

This binding is same as that of WSHttpBinding, except it supports duplex service. Duplex service is a service which uses duplex message pattern, which allows service to communicate with client via callback.

In WSDualHttpBinding reliable sessions are enabled by default. It also supports communication via SOAP intermediaries.

### WSFederationHttpBinding

This binding support federated security. It helps implementing federation which is the ability to flow and share identities across multiple enterprises or trust domains for authentication and authorization. It supports WS-Federation protocol.

### NetTcpBinding

This binding provides secure and reliable binding environment for .Net to .Net cross machine communication. By default it creates communication stack using WS-ReliableMessaging protocol for reliability, TCP for message delivery and windows security for message and authentication at run time. It uses TCP protocol and provides support for security, transaction and reliability.

### NetNamedPipeBinding

This binding provides secure and reliable binding environment for on-machine cross process communication. It uses NamedPipe protocol and provides full support for SOAP security, transaction and reliability. By default it creates communication stack with WS-ReliableMessaging for reliability, transport security for transfer security, named pipes for message delivery and binary encoding.

### NetMsmqBinding

* This binding provides secure and reliable queued communication for cross-machine environment.
* Queuing is provided by using MSMQ as transport.
* It enables for disconnected operations, failure isolation and load leveling

### NetPeerTcpBinding

* This binding provides secure binding for peer-to-peer environment and network applications.
* It uses TCP protocol for communication
* It provides full support for SOAP security, transaction and reliability.

# Binding configuration

Binding can be configured either through configuration file or Programming. Let us see the binding representation in each method.

### Administrative (Configuration file):

In the configuration file of the hosting application, you can add the <bindings> element inside the <system.serviceModel> element and add the properties to particular binding type. Properties corresponding to the particular binding type can be mentioned below. Name of the binding properties that you are going to use has to be mention in the end point.

<system.serviceModel>

<services>

<service name="MyService">

<endpoint address="http://localhost/IISHostedService/MyService.svc"

binding="wsHttpBinding" bindingName="wshttpbind" contract="IMyService">

<identity>

<dns value="localhost"/>

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

</service>

</services>

<bindings>

<wsHttpBinding>

<binding name="wshttpbind" allowCookies="true" closeTimeout="00:01:00"

receiveTimeout="00:01:00" />

</wsHttpBinding>

</bindings>

</system.serviceModel>

### Programming Model:

In the following code, I have created the *WSHttpBinding* object and assign the properties which to be configured. This binding object is added to the Service endpoint for client communication. Similarly you can also create any type of binding and add to endpoint.

//Create a URI to serve as the base address

Uri httpUrl = new Uri("http://localhost:8090/MyService/SimpleCalculator");

//Create ServiceHost

ServiceHost host =

new ServiceHost(typeof(MyCalculatorService.SimpleCalculator), httpUrl);

//Create Binding to add to end point

WSHttpBinding wshttpbind = new WSHttpBinding();

wshttpbind.AllowCookies = true;

wshttpbind.CloseTimeout = new TimeSpan(0, 1, 0);

wshttpbind.ReceiveTimeout = new TimeSpan(0, 1, 0);

//Add a service endpoint

host.AddServiceEndpoint

(typeof(MyCalculatorService.ISimpleCalculator), wshttpbind, "");

//Enable metadata exchange

ServiceMetadataBehavior smb = new ServiceMetadataBehavior();

smb.HttpGetEnabled = true;

host.Description.Behaviors.Add(smb);

//Start the Service

host.Open();

Console.WriteLine("Service is host at " + DateTime.Now.ToString());

Console.WriteLine("Host is running... Press key to stop");

Console.ReadLine();

**Note:** It is always good if you configure the binding properties using configuration file, because while moving to the production you no need to change in the code and recompile it. It is always good practice to represent in the configuration file.

**Service Contract**

Service contract describes the operation that service provide. A Service can have more than one service contract but it should have at least one Service contract.

Service Contract can be define using [ServiceContract] and [OperationContract] attribute. [ServiceContract] attribute is similar to the [WebServcie] attribute in the WebService and [OpeartionContract] is similar to the [WebMethod] in WebService.

* It describes the client-callable operations (functions) exposed by the service
* It maps the interface and methods of your service to a platform-independent description
* It describes message exchange patterns that the service can have with another party. Some service operations might be one-way; others might require a request-reply pattern
* It is analogous to the element in WSDL

To create a service contract you define an interface with related methods representative of a collection of service operations, and then decorate the interface with the *ServiceContract* Attribute to indicate it is a service contract. Methods in the interface that should be included in the service contract are decorated with the *OperationContract* Attribute.

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

int Add(int num1, int num2);

}

Once we define Service contract in the interface, we can create implement class for this interface.

public class SimpleCalculator : ISimpleCalculator

{

public int Add(int num1, int num2)

{

return num1 + num2;

}

}

With out creating the interface, we can also directly created the service by placing Contract in the implemented class. But it is not good practice of creating the service

[ServiceContract()]

public class SimpleCalculator

{

[OperationContract()]

public int Add(int num1, int num2)

{

return num1 + num2;

}

}

# Data Contract

A data contract is a formal agreement between a service and a client that abstractly describes the data to be exchanged.

Data contract can be explicit or implicit. Simple type such as int, string etc has an implicit data contract. User defined object are explicit or Complex type, for which you have to define a Data contract using [DataContract] and [DataMember] attribute.

A data contract can be defined as follows:

* It describes the external format of data passed to and from service operations
* It defines the structure and types of data exchanged in service messages
* It maps a CLR type to an XML Schema
* t defines how data types are serialized and deserialized. Through serialization, you convert an object into a sequence of bytes that can be transmitted over a network. Through deserialization, you reassemble an object from a sequence of bytes that you receive from a calling application.
* It is a versioning system that allows you to manage changes to structured data

We need to include *System.Runtime.Serialization* reference to the project. This assembly holds the *DataContract* and *DataMember* attribute.

Create user defined data type called Employee. This data type should be identified for serialization and deserialization by mentioning with [DataContract] and [DataMember] attribute.

[ServiceContract]

public interface IEmployeeService

{

[OperationContract]

Employee GetEmployeeDetails(int EmpId);

}

[DataContract]

public class Employee

{

private string m\_Name;

private int m\_Age;

private int m\_Salary;

private string m\_Designation;

private string m\_Manager;

[DataMember]

public string Name

{

get { return m\_Name; }

set { m\_Name = value; }

}

[DataMember]

public int Age

{

get { return m\_Age; }

set { m\_Age = value; }

}

[DataMember]

public int Salary

{

get { return m\_Salary; }

set { m\_Salary = value; }

}

[DataMember]

public string Designation

{

get { return m\_Designation; }

set { m\_Designation = value; }

}

[DataMember]

public string Manager

{

get { return m\_Manager; }

set { m\_Manager = value; }

}

}

Implementation of the service class is shown below. In GetEmployee method we have created the Employee instance and return to the client. Since we have created the data contract for the Employee class, client will aware of this instance whenever he creates proxy for the service.

public class EmployeeService : IEmployeeService

{

public Employee GetEmployeeDetails(int empId)

{

Employee empDetail = new Employee();

//Do something to get employee details and assign to 'empDetail' properties

return empDetail;

}

}

### Client side

On client side we can create the proxy for the service and make use of it. The client side code is shown below.

protected void btnGetDetails\_Click(object sender, EventArgs e)

{

EmployeeServiceClient objEmployeeClient = new EmployeeServiceClient();

Employee empDetails;

empDetails = objEmployeeClient.GetEmployeeDetails(empId);

**MessageHeaderArray Attribute**

Consider the Message contract type definition as shown below.

[MessageContract]

public class Department

{

[MessageHeader]

public string DepartmentID;

[MessageHeader]

public string DepartmentName;

[MessageHeader]

public Employees Employee();

}

In this we are having array of Employee type as message header. When this converted to SOAP Header it looks as shown below.

<Department>

<DepartmentID>PRO1243</DepartmentID>

<DepartmentName>Production</DepartmentName>

<Employees>

<Employee>Sam</Employee>

<Employee>Ram</Employee>

<Employee>Raja</Employee>

</Employees>

</Department>

Suppose you want to show the all employee detail in same level. We can use MessageHeaderArray attribute which will serialize the array element independently. If you use the MessageHeaderArray attribute of Employees, SOAP message will look as shown below.

<Department>

<DepartmentID>PRO1243</DepartmentID>

<DepartmentName>Production</DepartmentName>

<Employee>Sam</Employee>

<Employee>Ram</Employee>

<Employee>Raja</Employee>

</Department>

**Note:** MessageHeaderArray Attribute is applicable only for Array, not for collection.

# Message Contract Properties

### ProtectionLevel

You can mention the *MessageHeader* or *MessageBodyMember* to be signed or Encrypted using *ProtectionLevel* property.

**Example**

using System.Net.Security;

[MessageContract]

public class EmployeeDetails

{

[MessageHeader(ProtectionLevel=ProtectionLevel.None)]

public string EmpID;

[MessageBodyMember(ProtectionLevel = ProtectionLevel.Sign )]

public string Name;

[MessageBodyMember(ProtectionLevel = ProtectionLevel.Sign )]

public string Designation;

[MessageBodyMember(ProtectionLevel=ProtectionLevel.EncryptAndSign)]

public int Salary;

}

In the above type definition, we have made the different protection level for body. But the protection level of the body is determind by the highest *ProtectionLevel* property. By default if you are not specifying the protection level it takes 'EncryptAndSign'. So it good if you specify minimum ProtectionLevel required.

### Name and Namespace:

SOAP representation of the message element can be change by mentioning Name and Namespace property of the Header and Body member. By default namespace is the same as the namespace of the service contract that the message is participating. In the below example, I have mention the Name property to the EmpID and Name.

[MessageContract]

public class EmployeeDetails

{

[MessageHeader(Name="ID")]

public string EmpID;

[MessageBodyMember(Name="EmployeeName")]

public string Name;

[MessageBodyMember()]

public string Designation;

[MessageBodyMember()]

public int Salary;

}

When SOAP message representation, its name is changed to ID and EmployeeName.

<EmployeeDetails>

<ID>45634</ID>

<EmployeeName>Sam</EmployeeName>

<Designation>Software Engineer</Designation>

<Salary>25000</Salary>

</EmployeeDetails>

### Order

The order of the body elements are alpehabetical by default. But you can control the order, usiing *Order* property in the *MessageBody* attribute.

[MessageContract]

public class EmployeeDetails

{

[MessageHeader()]

public string EmpID;

[MessageBodyMember(Order=2)]

public string Name;

[MessageBodyMember(Order=3)]

public string Designation;

[MessageBodyMember(Order=1)]

public int Salary;

}

**Fault Contract**

Service that we develop might get error in come case. This error should be reported to the client in proper manner. Basically when we develop managed application or service, we will handle the exception using try- catch block. But these exceptions handlings are technology specific.

In order to support interoperability and client will also be interested only, what wents wrong? not on how and where cause the error.

By default when we throw any exception from service, it will not reach the client side. WCF provides the option to handle and convey the error message to client from service using SOAP Fault contract.

Suppose the service I consumed is not working in the client application. I want to know the real cause of the problem. How I can know the error? For this we are having Fault Contract. Fault Contract provides documented view for error accorded in the service to client. This help as to easy identity the what error has accord. Let us try to understand the concept using sample example.

**Step 1:** I have created simple calculator service with Add operation which will throw general exception as shown below

//Service interface

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

int Add(int num1, int num2);

}

//Service implementation

public class SimpleCalculator : ISimpleCalculator

{

public int Add(int num1, int num2)

{

//Do something

throw new Exception("Error while adding number");

}

}

**Step 2:** On client side code. Exceptions are handled using try-Catch block. Even though I have capture the exception when I run the application. I got the message that exceptions are not handled properly.

try

{

MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy

= new MyCalculatorServiceProxy.MyCalculatorServiceProxy();

Console.WriteLine("Client is running at " + DateTime.Now.ToString());

Console.WriteLine("Sum of two numbers... 5+5 =" + proxy.Add(5, 5));

Console.ReadLine();

}

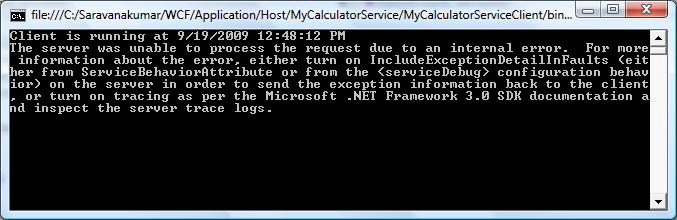
catch (Exception ex)

{

Console.WriteLine(ex.Message);

Console.ReadLine();

}



**Step 3:** Now if you want to send exception information form service to client, you have to use FaultException as shown below.

public int Add(int num1, int num2)

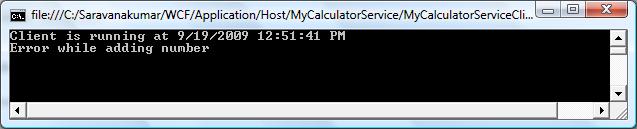
{

//Do something

throw new FaultException("Error while adding number");

}

**Step 4:** Output window on the client side is show below.



**Step 5:** You can also create your own Custom type and send the error information to the client using *FaultContract*. These are the steps to be followed to create the fault contract.

* Define a type using the data contract and specify the fields you want to return.
* Decorate the service operation with the FaultContract attribute and specify the type name.
* Raise the exception from the service by creating an instance and assigning properties of the custom exception.

**Step 6:** Defining the type using Data Contract

[DataContract()]

public class CustomException

{

[DataMember()]

public string Title;

[DataMember()]

public string ExceptionMessage;

[DataMember()]

public string InnerException;

[DataMember()]

public string StackTrace;

}

**Step 7:** Decorate the service operation with the *FaultContract*

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

[FaultContract(typeof(CustomException))]

int Add(int num1, int num2);

}

**Step 8:** Raise the exception from the service

public int Add(int num1, int num2)

{

//Do something

CustomException ex = new CustomException();

ex.Title = "Error Funtion:Add()";

ex.ExceptionMessage = "Error occur while doing add function.";

ex.InnerException = "Inner exception message from serice";

ex.StackTrace = "Stack Trace message from service.";

throw new FaultException(ex,"Reason: Testing the Fault contract") ;

}

**Step 9:** On client side, you can capture the service exception and process the information, as shown below.

try

{

MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy

= new MyCalculatorServiceProxy.MyCalculatorServiceProxy();

Console.WriteLine("Client is running at " + DateTime.Now.ToString());

Console.WriteLine("Sum of two numbers... 5+5 =" + proxy.Add(5, 5));

Console.ReadLine();

}

catch (FaultException<MyCalculatorService.CustomException> ex)

{

//Process the Exception

}

Security In WCF

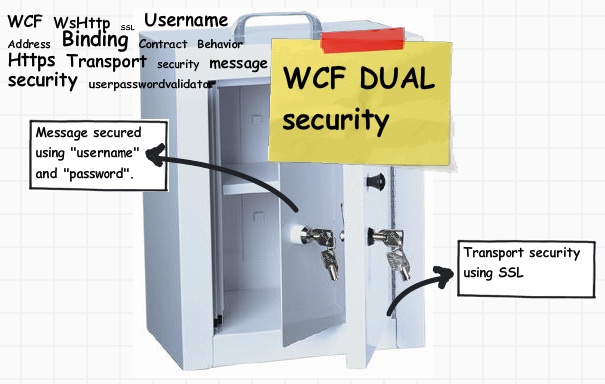
## Introduction and Goal

In the article, we will try to apply DUAL security using transport plus message on WCF services. So we will first try to understand the basic concepts of WCF security, i.e., transport and message. Once we understand the concept, we will move step by step into how to implement SSL and user name security on WCF services.

In case you are a complete fresher to WCF, you can start from [here](http://www.codeproject.com/KB/aspnet/WCF.aspx).

Watch my 500 videos on various topics like design patterns,WCF, WWF, WPF, LINQ, Silverlight, UML, Sharepoint, Azure,VSTS and a lot more [**here**](http://tinyurl.com/mra3hx). You can also view my WCF videos [**Part 1**](http://www.questpondvd.com/DemoVideos/WCF/SOA1/SOA1.php) and [**Part 2**](http://www.questpondvd.com/DemoVideos/WCF/SOA2/SOA2.php).

Enjoy my free ebook which covers major .NET related topics like WCF, WPF, WWF, Ajax, Core .NET, SQL Server, Architecture and lot more. Download from [**here**](http://tinyurl.com/4nvp9t)**.**

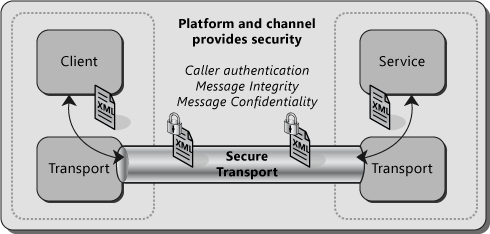


## 

# Transport Security

When using transport security, the user credentials and claims are passed by using the transport layer. In other words, user credentials are transport-dependent, which allows fewer authentication options compared to message security. Each transport protocol (TCP, IPC, MSMQ, or HTTP) has its own mechanism for passing credentials and handling message protection. The most common approach for this is to use Secure Sockets Layer (SSL) for encrypting and signing the contents of the packets sent over Secure HTTP (HTTPS).

Transport security is used to provide point-to-point security between the two endpoints (service and client). If there are intermediary systems between client and the service, each intermediate point must forward the message over a new SSL connection.



**Figure 1**   
Transport Security

Use transport security in the following scenarios:

* You are sending a message directly from your application to a WCF service and the message will not be routed through intermediate systems.
* Both the service and the client are located in an intranet.

Using transport security offers the following advantages:

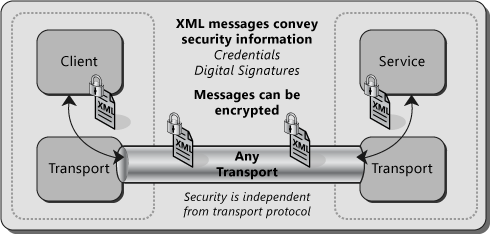
* It provides interoperability, meaning that communicating parties do not need to understand WS-Security specifications.
* It may result in better performance.
* Hardware accelerators can be used to further improve the performance.

Using transport security has the following disadvantages:

* Security is applied on a point-to-point basis, with no provision for multiple hops or routing through intermediate application nodes.
* It supports a limited set of credentials and claims compared to message security.
* It is transport-dependent upon the underlying platform, transport mechanism, and security service provider, such as NTLM or Kerberos.

# Message Security

When using message security, the user credentials and claims are encapsulated in every message using the WS-Security specification to secure messages. This option gives the most flexibility from an authentication perspective. You can use any type of security credentials you want, largely independent of transport, as long as both the client and service agree.



**Figure 2**   
Message Security

Use message security in the following scenarios:

* You are sending a message to a WCF service, and the message is likely to be forwarded to other WCF services or may be routed through intermediate systems.
* Your WCF clients are accessing the WCF service over the Internet and messages may be routed through intermediate systems.

Using message security offers the following advantages:

* It provides end-to-end security. Because message security directly encrypts and signs the message, having intermediaries does not break the security.
* It allows partial or selective message encryption and signing, thus improving overall application performance.
* Message security is transport-independent and therefore can be used with any transport protocol.
* It supports a wide set of credentials and claims, including the issue token that enables federated security.

Using message security has following disadvantages:

* This option may reduce performance compared to transport security because each individual message is encrypted and signed.
* It does not support interoperability with older ASMX clients, as it requires both the client and service to support WS-Security specifications.

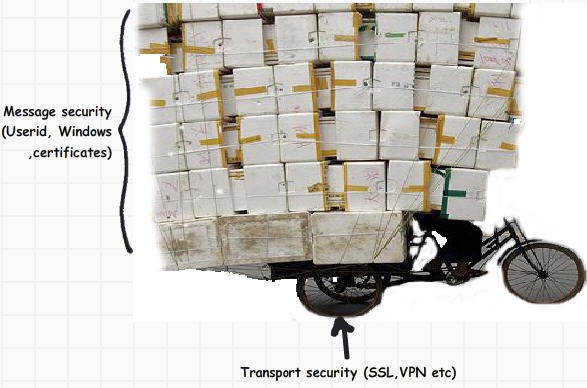
## Basics Transport and Message Level Security

On a broader basis, WCF supports two kinds of security - transport level and message level security. Transport means the medium on which WCF data travels while message means the actual data packets sent by WCF.

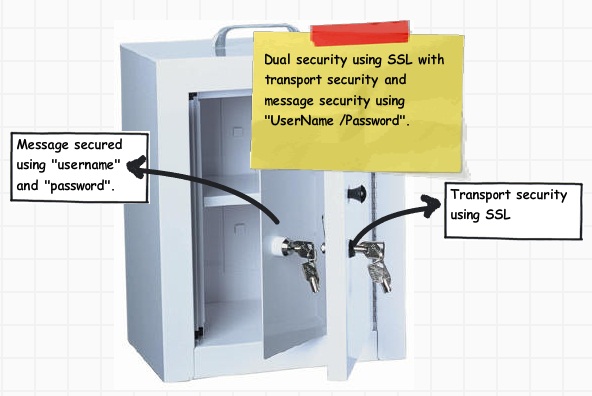
Transport medium can be protocols like TCP, HTTP, MSMQ, etc. These transport mediums by themselves provide security features like HTTP that can have SSL security (HTTPS). WCF has the capability of leveraging underlying transport security features on WCF service calls.

Message level security is provided in the data itself using WS-Security. In other words, it’s independent of the transport protocol. Some examples of message level security are messages encrypted using encryption algorithm, messages encrypted using X509 certificate etc, messages protected using username, etc.

WCF gives you an option to either just use message level security in standalone, transport level in standalone or combination of both. If you are interested in how to do message level security and transport security in a standalone manner, you can read more from here.



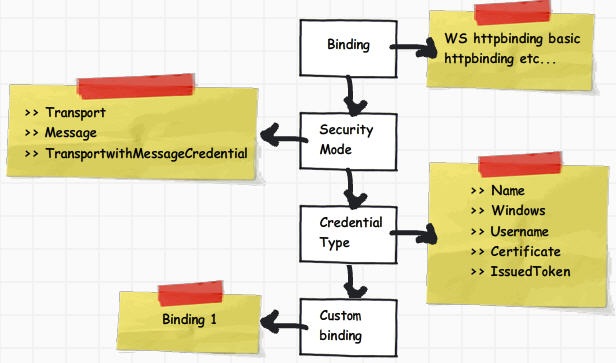
The best security is the combination of transport and message. In this article, we will see step by step how to implement dual security using ‘SSL’ plus message security using ‘Username’ using ‘WsHttpBinding’.



## Step 1: Customize ‘WsHttp’ Bindings with Security Mode and Credential Type

The first step is to customize your ‘Wshttp’ binding with proper security mode and credential type. There are three options in security mode ‘Transport’, ‘Message’ and ‘TransportWithMessageCredential’.

As we are implementing dual security, we need to use the last one, i.e., ‘TransportWithMessageCredential’ where the transport security is provided by SSL and message security is provided using ‘UserName and password’.



The second thing we need to provide is the credential type. There are five different credential types - none, windows, username, certificate and issued token. Credential type defines how the credentials will be passed over the transport layer. For the current instance, we will select ‘UserName’.

So summing up, we will provide security mode as ‘TransportWithMessageCredential’ and message security will be provided by ‘UserName’.

So create a WCF service using the WCF service template and in ‘web.config’ provide the security mode and credential type as shown in the below code snippet:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

<bindings>

<wsHttpBinding>

<binding name="Binding1">

<!-- UsernameToken over Transport Security -->

<security mode="TransportWithMessageCredential" >

<message clientCredentialType="UserName"/>

</security>

</binding>

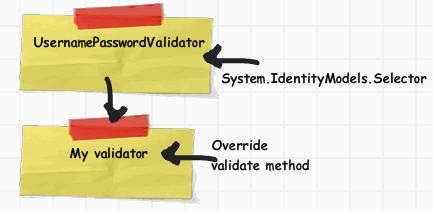
</wsHttpBinding>

</bindings>

## Step 2: Create your Custom Validator Class

Once we have customized the ‘WsHttp’ binding with security mode and credential type, it’s time to create the custom class which will do authentication of the user name provided.

In order to create your custom class, you need inherit the ‘UserNamePasswordValidator’ class which belongs to ‘System.IdentityModels.Selector’.



Below goes the code snippet of ‘MyValidator’ class. We need to override the ‘Validate’ method with the authentication logic as shown below:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

class MyValidator : UserNamePasswordValidator

{

public override void Validate(string userName, string password)

{

if ((userName == "shiv123") && (password == "pass123"))

{

}

else

{

throw new FaultException("Invalid credentials");

}

}

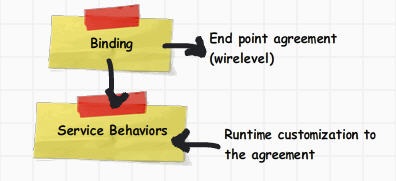
}

If the credentials are not proper, we have raised the ‘FaultException’ error which can be caught by the WCF client to display error messages.

## Step 3: Define Runtime Behavior

So we are almost 50% through now. We have customized the ‘WsHttp’ binding and created our custom class ‘MyValidator’ which will do the necessary authentication. The next step is to define behavior.

‘Behaviors’ define customized run time logic over the binding agreement. Currently we need to execute ‘MyValidator’ class logic for the ‘UserName’ provided in the WCF service by WCF client.



To specify the behavior, go to your ‘Web.config’ file and in the ‘servicecredentials’ tag, specify the ‘userNameAuthentication’ tag which points to the custom class ‘MyValidator’.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

<behaviors>

<serviceBehaviors>

<serviceCredentials>

<userNameAuthentication userNamePasswordValidationMode="Custom"

customUserNamePasswordValidatorType="MyValidator, app\_code"/>

</serviceCredentials>

</behavior>

## Step 4: Define SSL for your WCF Service

We have already mentioned transport security will be provided by SSL while message security will be provided by ‘username’. We have already configured ‘UserName’ message security using ‘MyValidator’ class which is specified in the behavior section of ‘web.config’ file. The next step is to configure SSL, i.e., transport security for our WCF service.

We will be using ‘makecert.exe’ which is a free tool given by Microsoft to enable HTTPS for testing purpose. MakeCert (Makecert.exe) is a command-line tool that creates an X.509 certificate that is signed by a system test root key or by another specified key. The certificate binds a certificate name to the public part of the key pair. The certificate is saved to a file, a system certificate store, or both.

You can get the same from “C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\Bin” or you can also get it from Windows SDK.

You can type the below thing through your DOS prompt on “C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\Bin”. Please note “compaq-jzp37md0” is the server name so you need to replace with your PC name.

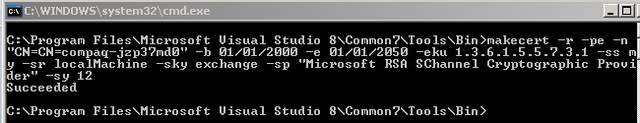
http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

makecert -r -pe -n "CN= compaq-jzp37md0 " -b 01/01/2000

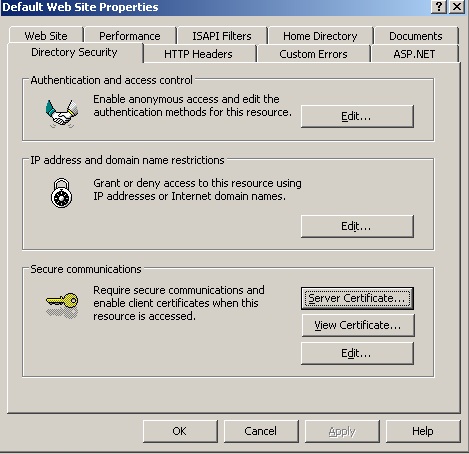
-e 01/01/2050 -eku 1.3.6.1.5.5.7.3.1 -ss my -sr

localMachine -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12

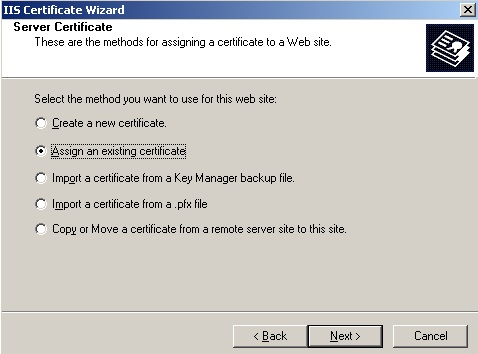
If you run the same through your command prompt, you should get a succeeded message as shown below:



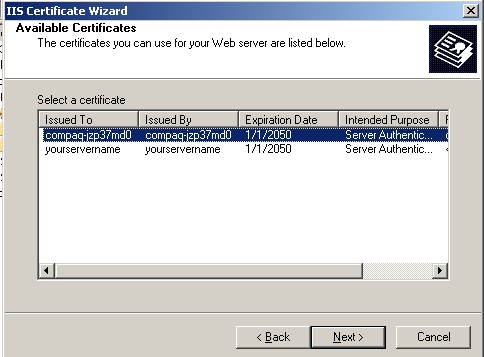
Now it’s time to assign this certificate to your IIS website. So go to IIS properties, click on directory security tab and you should see server certificate tab.



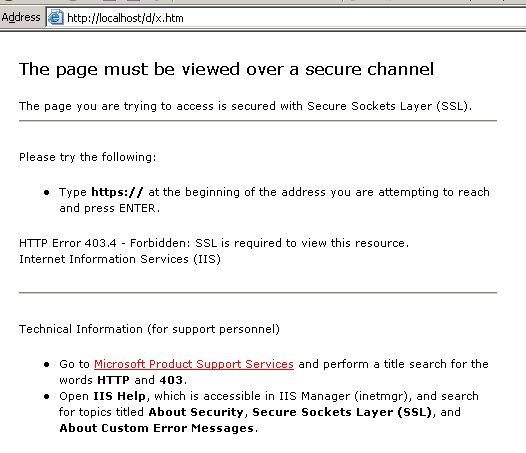
So click on the server certificate tab and you will then be walked through an IIS certificate wizard. Click ‘Assign an existing certificate’ from the wizard.



You can see a list of certificates. The “compaq-jzp37md0” certificate is the one which we just created using ‘makecert.exe’.



Now try to test the site without ‘https’ and you will get an error as shown below… That means your certificate is working.



Do not forget to enable IIS anonymous access.

We also need to make a couple of changes in the WCF service ‘Web.config’ ‘endpoint’ section as shown below. You can see how the address points to HTTPS and binding uses ‘mexHttpsBinding’.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

<service name="Service" behaviorConfiguration="ServiceBehavior">

<endpoint address=https://localhost/Service.svc

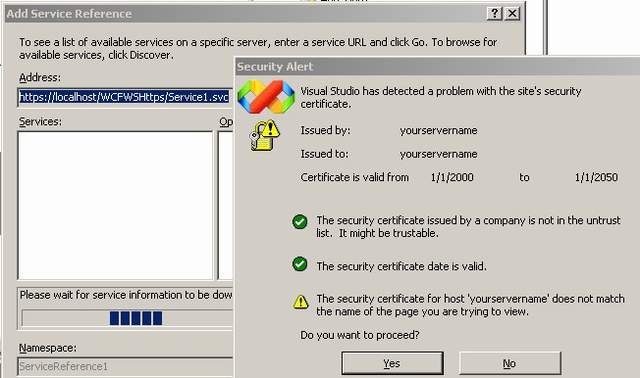
binding="wsHttpBinding" contract="IService" bindingConfiguration="Binding1">

<endpoint address="mex" binding="mexHttpsBinding" contract="IMetadataExchange"/>

</service>

## Step 5: Consume WCF Service

It’s time to consume WCF the service application. So click on add service reference and specify your service URL. You will be shown a warning box as shown in the below figure as the certificate is a test certificate. So just let it go.



The next step is to create WCF proxy client object and pass the credentials as shown in the below snippet:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

ServicePointManager.ServerCertificateValidationCallback =

new RemoteCertificateValidationCallback(IgnoreCertificateErrorHandler);

ServiceReference1.ServiceClient obj = new ServiceReference1.ServiceClient();

obj.ClientCredentials.UserName.UserName = "shiv123";

obj.ClientCredentials.UserName.Password = "pass123";

Response.Write(obj.GetData(12));

‘makecert.exe’ creates test certificates. In other words, it’s not signed by CA. So we need to suppress those errors in our ASP.NET client consumer. So we have created a function called as ‘IgnoreCertificateErrorHandler’ which returns true even if there are errors. This function is attached as a callback to ‘ServicePointManager.ServerCertificateValidationCallback’ as shown in the above code snippet.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/82737/6-Steps-to-Implement-DUAL-Security-on-WCF-using-Us)

public static bool IgnoreCertificateErrorHandler

(object sender, X509Certificate certificate, X509Chain chain,

SslPolicyErrors sslPolicyErrors)

{

return true;

}

## Step 6: Run your WCF Service

If everything goes appropriately, you should be able to run the WCF service. Try changing the use rid and password to something else you should get the fault exception message provided in the ‘MyValidator’ class.

[**WCF Username Authentication**](http://www.devatwork.nl/2007/05/wcf-username-authentication/)

May 31st, 2007 by [Trilobyte](http://www.devatwork.nl/author/trilobyte/)  
Tags:[WCF](http://www.devatwork.nl/tag/wcf/)

In this post I will explain how you can build an Windows Communication Foundation web service and client which use a Username and Password combination to authenticate a user. The most difficult action is to create a X509 certificate which is used to encrypt messages passed back and forward to the server.

In this application we will use WCF’s wsHttpBinding and message level security provided by an X509 certificate. The X509 certificate encryption is required by WCF because the client credentials (username/password) are passed as clear text in the SOAP message.

There is one problem that we will face during this series of posts. WCF is reluctant to accept a test certificate, it requires a lot of extra work to get it done. However once you understand the steps that you need to take, you will find it an repetitive but easy task.

I hope you find this post useful. If you have any questions or comments, feel free to post them as reactions on this post. Enjoy!

**Generating a Certificate**

The first step is to create a test X509 certificate, which is used to encrypt the messages. The certificate will be placed in the ‘My’ folder on the ‘Local Machine’ store under the name ‘MyServerCert’.

To generate the necessary certificate, execute the following command in the windows SDK command line utility:

|  |  |
| --- | --- |
| 1 | makecert.exe -sr LocalMachine -ss My -a sha1 -n CN=MyServerCert -sky exchange –pe |

**Warning**: This certificate should be used for testing purposes only.

**Setting Up the Service**

There are several steps we need to perform on the service, to force it to use username/password validation.

The first step is to implement the validator class which takes the username/password combination and ensures they are correct. First make sure you reference the System.IdentityModel assembly, next create a new class and derive it from System.IdentityModel.Selectors.UserNamePasswordValidator. Now override the Validate function derived from UserNamePasswordValidator.

Now you can write code that checks if the username/password combination is valid, if the combination is not valid throw a System.IdentityModel.Tokens.SecurityTokenException.

Here is my implementation:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class UsernameValidator: UserNamePasswordValidator  {  public override void Validate(string userName, string password)      {          // validate arguments          if (string.IsNullOrEmpty(userName))              throw new ArgumentNullException("userName");          if (string.IsNullOrEmpty(password))              throw new ArgumentNullException("password");            // check if the user is not test          if (userName != "test" || password != "test")              throw new SecurityTokenException("Unknown username or password");      }  } |

The next step is to configure the service to use our custom validator and enforce the username/password client credentials.

First create a new binding with the following configuration:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <bindings>  <wshttpbinding>          <binding name="mySecureBinding">              <security mode="Message">                  <message clientCredentialType="UserName" />              </security>          </binding>      </wshttpbinding>  </bindings> |

Notice that the binding enforces message level security and the client has to provide the UserName credentials.

Now that the binding is ready we need to define some behaviour for the service endpoint:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <behaviors>  <servicebehaviors>          <behavior name="defaultProfile">              <servicecredentials>                  <servicecertificate findValue="MyServerCert" x509FindType="FindBySubjectName" storeLocation="LocalMachine" storeName="My" />                  <usernameauthentication userNamePasswordValidationMode="Custom" customUserNamePasswordValidatorType="Premotion.Services.UsernameValidator, App\_Code" />              </servicecredentials>          </behavior>      </servicebehaviors>  </behaviors> |

There are two interesting elements in this section: serviceCertificate and userNameAuthentication. The first specifies the certificate which the service uses to encrypt and decrypt the messages. The second element specifies our custom validator.

Now make sure all your endpoints use the correct binding and behaviour. That’s all for the server!

**Modifying the Client**

Modifying the client could have been an easy task, if we had accesses to a valid certificate. WCF will not accept the test certificate without a bunch of tricks. First I will pretend the service certificate is valid, then I will explain how to get WCF to accept the test certificate.

The first step is to create a new binding in the application configuration:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <bindings>  <wshttpbinding>          <binding name="mySecureBinding">              <security mode="Message">                  <message clientCredentialType="UserName" />              </security>          </binding>      </wshttpbinding>  </bindings> |

This exactly the same binding as we configured the service with.

The next is to set the username/password credentials in code. The following code should be placed next to the code which instantiates the service client:

|  |  |
| --- | --- |
| 1  2 | base.ClientCredentials.UserName.UserName    = "test";  base.ClientCredentials.UserName.Password    = "test"; |

That is all. If you have a valid certificate you can run the code, if you have the test certificate perform the steps below to get the code running.

**Bypass Certificate Validation**

First make sure you reference the System.IdentityModel assembly. The first step is to create a class which validates the certificates and derive it from System.IdentityModel.Selectors.X509CertificateValidator. Override the Validate method. You can leave the implementation empty if you want all the certificates to pass. If you detect a wrong certificate you can throw a System.IdentityModel.Tokens.SecurityTokenValidationException.

Here is my implementation:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class MyX509Validator: X509CertificateValidator  {      public override void Validate(X509Certificate2 certificate)      {          // validate argument          if (certificate == null)              throw new ArgumentNullException("certificate");            // check if the name of the certifcate matches          if (certificate.SubjectName.Name != "CN=MyServerCert")              throw new SecurityTokenValidationException("Certificated was not issued by thrusted issuer");      }  } |

The next step is to create a new endpoint behaviour, which tells WCF to use our custom certificate validator:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <behaviors>  <endpointbehaviors>          <behavior name="myClientBehavior">              <clientcredentials>                  <servicecertificate>                      <authentication certificateValidationMode="Custom" customCertificateValidatorType="Premotion.Services.MyX509Validator,client" />                  </servicecertificate>              </clientcredentials>          </behavior>      </endpointbehaviors>  </behaviors> |

The last step is to set the DNS identity for the endpoint:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <client>      <endpoint address="<http://localhost:1494/services/coreservice.svc>" binding="wsHttpBinding" bindingConfiguration="mySecureBinding" contract="Premotion.Services.ICoreServiceClientContact" name="CoreService" behaviorConfiguration="myClientBehavior">          <identity>              <dns value="MyServerCert"/>          </identity>      </endpoint>  </client> |

Now everything is finished and you can execute the code.

You can download the sample by clicking [here](http://www.devatwork.nl/wp-content/uploads/wcfusername.zip), make sure you create the certificate manually! I hope you gained some insight on how WCF works with username/password authentication. Thank you for reading, if you have any questions or comments post them

**WCF Interview Questions and Answer**

**Q1. What is WCF?**  
WCF stands for Windows Communication Foundation. It is a Software development kit for developing services on Windows. WCF is introduced in .NET 3.0. in the System.ServiceModel namespace. WCF is based on basic concepts of Service oriented architecture (SOA)

Difference between XML Seerialization and Datacontract serialization:

A practical benefit of the design of the DataContractSerializer is better performance over Xmlserializer.  
\* XML Serialization does not indicate the which fields or properties of the type are serialized into XML where as DataCotratSerializer Explicitly shows the which fields or properties are serialized into XML.  
\* The DataContractSerializer can translate the HashTable into XML.  
  
**Q2. What is endpoint in WCF service?**  
The endpoint is an Interface which defines how a client will communicate with the service. It consists of three main points: Address,Binding and Contract.  
  
**Q3. Explain Address,Binding and contract for a WCF Service?**  
Address:Address defines where the service resides.  
Binding:Binding defines how to communicate with the service.  
Contract:Contract defines what is done by the service.  
  
**Q4. What are the various address format in WCF?**  
a)HTTP Address Format:--> http://localhost:  
b)TCP Address Format:--> net.tcp://localhost:  
c)MSMQ Address Format:--> net.msmq://localhost:  
  
**Q5. What are the types of binding available in WCF?**  
A binding is identified by the transport it supports and the encoding it uses. Transport may be HTTP,TCP etc and encoding may be text,binary etc. The popular types of binding may be as below:  
a)BasicHttpBinding  
b)NetTcpBinding  
c)WSHttpBinding  
d)NetMsmqBinding  
 **Q6. What are the types of contract available in WCF?**  
The main contracts are:  
a)Service Contract:Describes what operations the client can perform.  
b)Operation Contract : defines the method inside Interface of Service.  
c)Data Contract:Defines what data types are passed  
d)Message Contract:Defines wheather a service can interact directly with messages  
  
**Q7. What are the various ways of hosting a WCF Service?**  
a)IIS b)Self Hosting c)WAS (Windows Activation Service)  
 **Q8. WWhat is the proxy for WCF Service?**  
A proxy is a class by which a service client can Interact with the service.  
By the use of proxy in the client application we are able to call the different methods exposed by the service  
 **Q9. How can we create Proxy for the WCF Service?**  
We can create proxy using the tool svcutil.exe after creating the service.  
We can use the following command at command line.  
svcutil.exe \*.wsdl \*.xsd /language:C# /out:SampleProxy.cs /config:app.config  
  
**Q10.What is the difference between WCF Service and Web Service?**  
a)WCF Service supports both http and tcp protocol while webservice supports only http protocol.  
b)WCF Service is more flexible than web service.  
  
**Q11.What is DataContract and ServiceContract?Explain**   
Data represented by creating DataContract which expose the  
data which will be transefered /consumend from the serive  
to its clients.  
  
\*\*Operations which is the functions provided by this  
service.  
  
To write an operation on WCF,you have to write it as an  
interface,This interface contains the "Signature" of the  
methods tagged by ServiceContract attribute,and all methods  
signature will be impelemtned on this interface tagged with  
OperationContract attribute.  
  
and to implement these serivce contract you have to create  
a class which implement the interface and the actual  
implementation will be on that class.  
  
  
Code Below show How to create a Service Contract:

**Code:**

[ServiceContract]  
Public Interface IEmpOperations  
{  
[OperationContract]  
Decimal Get EmpSal(int EmpId);  
  
}  
  
Class MyEmp: IEmpOperations  
{  
Decimal Get EmpSal()  
{  
// Implementation of this method.  
}  
}

1. What is the difference between WCF and ASMX Web Services?

Simple and basic difference is that ASMX or ASP.NET web service is designed to send and receive messages using SOAP over HTTP only. While WCF can exchange messages using any format (SOAP is default) over any transport protocol (HTTP, TCP/IP, MSMQ, NamedPipes etc).

Another tutorial [WCF Vs ASMX](http://wcfanswers.blogspot.com/2012/06/wcf-vs-asmx-web-services.html) has detailed discussion on it.

2. What are WCF Service Endpoints? Explain.

For **Windows Communication Foundation** services to be consumed, it’s necessary that it must be exposed; Clients need information about service to communicate with it. This is where service endpoints play their role.

A **WCF service** endpoint has three basic elements i.e. Address, Binding and Contract.

* **Address:** It defines "WHERE". Address is the URL that identifies the location of the service.
* **Binding:** It defines "HOW". Binding defines how the service can be accessed.
* **Contract:** It defines "WHAT". Contract identifies what is exposed by the service.

3. What are the possible ways of hosting a WCF service? Explain.

For a **Windows Communication Foundation** service to host, we need at least a managed process, a ServiceHost instance and an Endpoint configured. Possible approaches for hosting a service are:

1. Hosting in a Managed Application/ Self Hosting
   1. Console Application
   2. Windows Application
   3. Windows Service
2. Hosting on Web Server
   1. IIS 6.0 (ASP.NET Application supports only HTTP)
   2. Windows Process Activation Service (WAS) i.e. IIS 7.0 supports HTTP, TCP, NamedPipes, MSMQ.

4. How we can achieve Operation Overloading while exposing WCF Services?

By default, WSDL doesn’t support operation overloading. Overloading behavior can be achieved by using "Name" property of OperationContract attribute.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

[ServiceContract]

interface IMyCalculator

{

[OperationContract(Name = "SumInt")]

int Sum(int arg1,int arg2);

[OperationContract(Name = "SumDouble")]

double Sum(double arg1,double arg2);

}

When the proxy will be generated for these operations, it will have 2 methods with different names i.e. SumInt and SumDouble.

**5. What Message Exchange Patterns (MEPs) supported by WCF? Explain each of them briefly.**

1. Request/Response 2. One Way 3. Duplex

**Request/Response**

It’s the default pattern. In this pattern, a response message will always be generated to consumer when the operation is called, even with the void return type. In this scenario, response will have empty SOAP body.

**One Way**

In some cases, we are interested to send a message to service in order to execute certain business functionality but not interested in receiving anything back. OneWay MEP will work in such scenarios. If we want queued message delivery, OneWay is the only available option.

**Duplex**

The Duplex MEP is basically a two-way message channel. In some cases, we want to send a message to service to initiate some longer-running processing and require a notification back from service in order to confirm that the requested process has been completed.

6. What is DataContractSerializer and How its different from XmlSerializer?

Serialization is the process of converting an object instance to a portable and transferable format. So, whenever we are talking about **web services**, serialization is very important.

Windows Communication Foundation has DataContractSerializer that is new in .NET 3.0 and uses opt-in approach as compared to XmlSerializer that uses opt-out. Opt-in means specify whatever we want to serialize while Opt-out means you don’t have to specify each and every property to serialize, specify only those you don’t want to serialize. DataContractSerializer is about 10% faster than XmlSerializer but it has almost no control over how the object will be serialized. If we wanted to have more control over how object should be serialized that XmlSerializer is a better choice.

7. How we can use MessageContract partially with DataContract for a service operation in WCF?

MessageContract must be used all or none. If we are using MessageContract into an operation signature, then we must use MessageContract as the only parameter type and as the return type of the operation.

8. Which standard binding could be used for a service that was designed to replace an existing ASMX web service?

The basicHttpBinding standard binding is designed to expose a service as if it is an ASMX/ASP.NET web service. This will enable us to support existing clients as applications are upgrade to WCF.

9. Please explain briefly different Instance Modes in WCF?

WCF will bind an incoming message request to a particular service instance, so the available modes are:

* **Per Call**: instance created for each call, most efficient in term of memory but need to maintain session.
* **Per Session**: Instance created for a complete session of a user. Session is maintained.
* **Single**: Only one instance created for all clients/users and shared among all.Least efficient in terms of memory.

10. Please explain different modes of security in WCF? Or Explain the difference between Transport and Message Level Security.

In Windows Communication Foundation, we can configure to use security at different levels

**a. Transport Level security** means providing security at the transport layer itself. When dealing with security at Transport level, we are concerned about integrity, privacy and authentication of message as it travels along the physical wire. It depends on the binding being used that how WCF makes it secure because most of the bindings have built-in security.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<netTcpBinding>

<binding name="netTcpTransportBinding">

<security mode="Transport">

<Transport clientCredentialType="Windows" />

</security>

</binding>

</netTcpBinding>

**b. Message Level Security** For Tranport level security, we actually ensure the transport that is being used should be secured but in message level security, we actually secure the message. We encrypt the message before transporting it.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<wsHttpBinding>

<binding name="wsHttpMessageBinding">

<security mode="Message">

<Message clientCredentialType="UserName" />

</security>

</binding>

</wsHttpBinding>

It totally depends upon the requirements but we can use a mixed security mode also as follows:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<basicHttpBinding>

<binding name="basicHttp">

<security mode="TransportWithMessageCredential">

<Transport />

<Message clientCredentialType="UserName" />

</security>

</binding>

</basicHttpBinding>

**Socond example of security in wcf:**

**Implementing Transport Message Security with WCF 4.0 and VS 2010 RC**

I hope most of you have [downloaded VS 2010 RC](http://www.microsoft.com/visualstudio/en-us/download) by now and have started exploring its new features. I have received feedback in the past where developers have faced challenges in implementing security for WCF services in their applications. One of the nice features of WCF Security is that we can configure it using a custom user name – password validation mechanism.  In this two part series, I will explain how to create a certificate, create a WCF service configured to use this certificate and host the WCF service on IIS 7.5 with SSL. In this article, I will show you how to create a certificate and create a WCF service which uses this certificate. In this next part of this article, I will demonstrate how to host this WCF Service on IIS 7.5 with SSL.

I am developing this solution on Windows 7 Enterprise edition.

**Creating X509 Certificate**

**Step 1:** Since we need to use SSL for transport security, a certificate must be created and configured with the WCF service. To create a certificate, follow the steps shown below:

i)             Open command prompt of VS2010 and write the following command for creating a certificate:

makecert.exe -sr LocalMachine -ss My -a sha1 -n CN=MyCustomUsrPwdCert –sky exchange –pe

(Note: If you want to know more about this command you can visit [this](http://msdn.microsoft.com/en-us/library/bfsktky3(VS.80).aspx) link)

ii)            You can view this certificate created in the certificate store. For this purpose, you need to use the Microsoft Management Console (MMC).

iii)           This certificate should be available in the ‘Personal’ store of the ‘LocalMachine’ account so that it can be configured for the WCF service. (Note: You need to use either ‘CertMgr’ command line tool or MMC for import and export certificates.)

**Creating WCF Service using SSL**

**Step 2:** Open VS2010 and create a blank solution, name it as ‘WCF\_SecurityPractice’.

**Step 3:** To this solution, add a WCF service application project and name it as ‘WCF\_CustomUNamePwdeSecureService’. Rename ‘IService1.cs’ to ‘IService.cs’, rename ‘Service1.cs’ to ‘Service.cs’.

**Step 4:** Add the following code in ‘Iservice.cs’

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace WCF\_CustomUNamePwdeSecureService

{

    [ServiceContract]

    public interface IService

    {

        [OperationContract]

        List<clsEmployee> GetAllEmployees();

    }

    [DataContract]

    public class clsEmployee

    {

        [DataMember]

        public int EmpNo { get; set; }

        [DataMember]

        public string EmpName { get; set; }

    }

}

VB.NET

Imports System

Imports System.Collections.Generic

Imports System.Linq

Imports System.Runtime.Serialization

Imports System.ServiceModel

Imports System.ServiceModel.Web

Imports System.Text

Namespace WCF\_CustomUNamePwdeSecureService

      <ServiceContract> \_

      Public Interface IService

            <OperationContract> \_

            Function GetAllEmployees() As List(Of clsEmployee)

      End Interface

      <DataContract> \_

      Public Class clsEmployee

            Private privateEmpNo As Integer

            <DataMember> \_

            Public Property EmpNo() As Integer

                  Get

                        Return privateEmpNo

                  End Get

                  Set(ByVal value As Integer)

                        privateEmpNo = value

                  End Set

            End Property

            Private privateEmpName As String

            <DataMember> \_

            Public Property EmpName() As String

                  Get

                        Return privateEmpName

                  End Get

                  Set(ByVal value As String)

                        privateEmpName = value

                  End Set

            End Property

      End Class

End Namespace

**Step 5:** Write the following code in ‘Service.cs’. This code implements ‘IService’ interface in ‘Service’ class.

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using System.Security;

namespace WCF\_CustomUNamePwdeSecureService

{

    public class Service : IService

    {

        public List<clsEmployee> GetAllEmployees()

        {

            List<clsEmployee> lstEmp = null;

            if (OperationContext.Current.ServiceSecurityContext.PrimaryIdentity.IsAuthenticated == false)

            {

                throw new SecurityException();

            }

            else

            {

                lstEmp = new List<clsEmployee>()

                {

                    new clsEmployee() {EmpNo=101,EmpName="Mahesh 1"},

                    new clsEmployee() {EmpNo=102,EmpName="Mahesh 2"},

                    new clsEmployee() {EmpNo=103,EmpName="Mahesh 3"},

                    new clsEmployee() {EmpNo=104,EmpName="Mahesh 4"},

                    new clsEmployee() {EmpNo=105,EmpName="Mahesh 5"},

                    new clsEmployee() {EmpNo=106,EmpName="Mahesh 6"},

                    new clsEmployee() {EmpNo=107,EmpName="Mahesh 7"},

                    new clsEmployee() {EmpNo=108,EmpName="Mahesh 8"},

                    new clsEmployee() {EmpNo=109,EmpName="Mahesh 9"},

                    new clsEmployee() {EmpNo=1010,EmpName="Mahesh 10"}

                };

            }

            return lstEmp;

        }

    }

}

VB.NET

Imports System

Imports System.Collections.Generic

Imports System.Linq

Imports System.Runtime.Serialization

Imports System.ServiceModel

Imports System.ServiceModel.Web

Imports System.Text

Imports System.Security

Namespace WCF\_CustomUNamePwdeSecureService

      Public Class Service

            Implements IService

            Public Function GetAllEmployees() As List(Of clsEmployee)

                  Dim lstEmp As List(Of clsEmployee) = Nothing

                  If OperationContext.Current.ServiceSecurityContext.PrimaryIdentity.IsAuthenticated = False Then

                        Throw New SecurityException()

                  Else

                        lstEmp = New List(Of clsEmployee) (New clsEmployee() {New clsEmployee() With {.EmpNo=101, .EmpName="Mahesh 1"}, New clsEmployee() With {.EmpNo=102, .EmpName="Mahesh 2"}, New clsEmployee() With {.EmpNo=103, .EmpName="Mahesh 3"}, New clsEmployee() With {.EmpNo=104, .EmpName="Mahesh 4"}, New clsEmployee() With {.EmpNo=105, .EmpName="Mahesh 5"}, New clsEmployee() With {.EmpNo=106, .EmpName="Mahesh 6"}, New clsEmployee() With {.EmpNo=107, .EmpName="Mahesh 7"}, New clsEmployee() With {.EmpNo=108, .EmpName="Mahesh 8"}, New clsEmployee() With {.EmpNo=109, .EmpName="Mahesh 9"}, New clsEmployee() With {.EmpNo=1010, .EmpName="Mahesh 10"}})

                  End If

                  Return lstEmp

            End Function

      End Class

End Namespace

**Step 6:** Now since the service is going to authenticate caller using ‘UserName’ and ‘Password’, we need to write a custom validator. To do this, follow the steps shown below;

         Right click on the WCF service project and add reference of ‘System.IdentityModel’ namespace.

         If you have a SQL Server Database. create a table of name ‘Users’ with following Schema:

         To the project, add a class, name it as ‘AuthenticationHelper’ and write the following code:

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IdentityModel;

using System.IdentityModel.Selectors;

using System.Data.SqlClient;

using System.Data;

namespace WCF\_CustomUNamePwdeSecureService

{

    public class AuthenticationHelper : UserNamePasswordValidator

    {

        SqlConnection Conn;

        SqlCommand Cmd;

        SqlDataReader Reader;

        public override void Validate(string userName, string password)

        {

            if (userName == null || password == null)

            {

                throw new Exception("User Name or Password cannot be null");

            }

            if (!this.CheckIfUserNameExist(userName))

            {

                throw new Exception("Sorry!This User is Not Present");

            }

            if (!this.AuthenticateUser(userName, password))

            {

                throw new Exception("Invalid User Name or Password");

            }

        }

        #region Service Method

        //Following Method check wheather UserName is present in table or not

        private bool CheckIfUserNameExist(string userName)

        {

            bool Exists = false;

            Conn = new SqlConnection("Data Source=.\\dbserver;Initial Catalog=Company;Integrated Security=SSPI");

            Cmd = new SqlCommand();

            Conn.Open();

            Cmd.CommandText = "Select UserName from Users where UserName=@UserName";

            Cmd.Connection = Conn;

            Cmd.Parameters.AddWithValue("@UserName", userName);

            Reader = Cmd.ExecuteReader();

            DataTable DtUser = new DataTable();

            DtUser.Load(Reader);

            int Count = DtUser.Rows.Count; //Check the Count of Rows

            if (Count != 0) //If row count is <> 0 then user exists

                Exists = true;

            Conn.Close();

            return Exists;

        }

        private bool AuthenticateUser(string userName, string password)

        {

            bool Valid = false;

            Conn = new SqlConnection("Data Source=.\\dbserver;Initial Catalog=Company;Integrated Security=SSPI");

            Cmd = new SqlCommand();

            Conn.Open();

            Cmd.CommandText = "Select Password from Users where UserName=@UserName";

            Cmd.Connection = Conn;

            Cmd.Parameters.AddWithValue("@UserName", userName.Trim());

            Reader = Cmd.ExecuteReader();

            Reader.Read();

            if (Reader["Password"].ToString() == password.Trim())

                Valid = true;

            Conn.Close();

            return Valid;

        }

        #endregion

    }

}

VB.NET

Imports System

Imports System.Collections.Generic

Imports System.Linq

Imports System.Text

Imports System.IdentityModel

Imports System.IdentityModel.Selectors

Imports System.Data.SqlClient

Imports System.Data

Namespace WCF\_CustomUNamePwdeSecureService

      Public Class AuthenticationHelper

            Inherits UserNamePasswordValidator

            Private Conn As SqlConnection

            Private Cmd As SqlCommand

            Private Reader As SqlDataReader

            Public Overrides Sub Validate(ByVal userName As String, ByVal password As String)

                  If userName Is Nothing OrElse password Is Nothing Then

                        Throw New Exception("User Name or Password cannot be null")

                  End If

                  If (Not Me.CheckIfUserNameExist(userName)) Then

                        Throw New Exception("Sorry!This User is Not Present")

                  End If

                  If (Not Me.AuthenticateUser(userName, password)) Then

                        Throw New Exception("Invalid User Name or Password")

                  End If

            End Sub

            #Region "Service Method"

            'Following Method check wheather UserName is present in table or not

            Private Function CheckIfUserNameExist(ByVal userName As String) As Boolean

                  Dim Exists As Boolean = False

                  Conn = New SqlConnection("Data Source=.\dbserver;Initial Catalog=Company;Integrated Security=SSPI")

                  Cmd = New SqlCommand()

                  Conn.Open()

                  Cmd.CommandText = "Select UserName from Users where UserName=@UserName"

                  Cmd.Connection = Conn

                  Cmd.Parameters.AddWithValue("@UserName", userName)

                  Reader = Cmd.ExecuteReader()

                  Dim DtUser As New DataTable()

                  DtUser.Load(Reader)

                  Dim Count As Integer = DtUser.Rows.Count 'Check the Count of Rows

                  If Count <> 0 Then 'If row count is <> 0 then user exists

                        Exists = True

                  End If

                  Conn.Close()

                  Return Exists

            End Function

            Private Function AuthenticateUser(ByVal userName As String, ByVal password As String) As Boolean

                  Dim Valid As Boolean = False

                  Conn = New SqlConnection("Data Source=.\dbserver;Initial Catalog=Company;Integrated Security=SSPI")

                  Cmd = New SqlCommand()

                  Conn.Open()

                  Cmd.CommandText = "Select Password from Users where UserName=@UserName"

                  Cmd.Connection = Conn

                  Cmd.Parameters.AddWithValue("@UserName", userName.Trim())

                  Reader = Cmd.ExecuteReader()

                  Reader.Read()

                  If Reader("Password").ToString() = password.Trim() Then

                        Valid = True

                  End If

                  Conn.Close()

                  Return Valid

            End Function

            #End Region

      End Class

End Namespace

The above class is inherited from ‘UserNamePasswordValidator’ class. This class provides ‘Validate’ method which needs to be overridden by using custom logic for validating user name and password. In this article, I am connecting to my SQL server 2008 instance to verify user name and password, passed by my client application.

**Step 7:** Now open the Web.config file and write the following configuration:

<?xmlversion="1.0"?>

<configuration>

 <system.web>

    <compilationdebug="true"targetFramework="4.0" />

 </system.web>

 <system.serviceModel>

    <services>

      <servicename="WCF\_CustomUNamePwdeSecureService.Service"behaviorConfiguration="MyBehavior">

        <endpointaddress=""

                  binding="basicHttpBinding"

                  contract="WCF\_CustomUNamePwdeSecureService.IService"

                   bindingConfiguration="MyBind">

          <identity>

            <dnsvalue="localhost"/>

          </identity>

        </endpoint>

        <endpointaddress="mex"binding="mexHttpsBinding"contract="IMetadataExchange"/>

      </service>

    </services>

    <bindings>

      <basicHttpBinding>

        <bindingname="MyBind">

          <securitymode="TransportWithMessageCredential">

            <messageclientCredentialType="UserName"/>

          </security>

        </binding>

      </basicHttpBinding>

    </bindings>

    <behaviors>

      <serviceBehaviors>

        <behaviorname="MyBehavior">

          <serviceCredentials>

            <serviceCertificatestoreName="My"storeLocation="LocalMachine"x509FindType="FindBySubjectName"findValue="MyCustomUsrPwdCert"/>

            <userNameAuthenticationuserNamePasswordValidationMode="Custom"

                                    customUserNamePasswordValidatorType="WCF\_CustomUNamePwdeSecureService.AuthenticationHelper,WCF\_CustomUNamePwdeSecureService"/>

          </serviceCredentials>

          <serviceMetadata/>

          <serviceDebugincludeExceptionDetailInFaults="False" />

        </behavior>

      </serviceBehaviors>

    </behaviors>

 </system.serviceModel>

 <system.webServer>

    <modulesrunAllManagedModulesForAllRequests="true"/>

 </system.webServer>

</configuration>

The above configuration is explained here:

         BasicHttpBinding is used so that the WCF service can be used by all types of clients including Silverlight 3.0.

         TransportWithMessageCredential is used to provide transport level security over Http using Security Socket Layer (SSL).

         In the Binding configuration, client credential types is defined to ‘UserName’. This demands that the consumer of the service must pass a valid user credential for establishing successful communication with WCF service.

         In the behavior configuration, the certificate which we have created is specified for encryption.

         ‘userNameAuthentication’ here defines what component is used to verify user name and password passed by the client application.

So now your WCF service is ready to be deployed

**What is WCF?**   
Windows Communication Foundation or just WCF is a programming framework used to build applications that communicate with each other. It is a part of the .NET Framework dedicated to communications.   
  
  
**What are the different WCF binding available?**   
• BasicHttpBinding   
• WSHttpBinding   
• WSDualHttpBinding   
• WSFederationHttpBinding   
• NetTcpBinding   
• NetNamedPipeBinding   
• NetMsmqBinding   
• NetPeerTcpBinding   
• MsmqIntegrationBinding   
  
  
**What is the proxy for WCF Service?**   
A proxy is a class by which a service client can Interact with the service. By the use of proxy in the client application we are able to call the different methods exposed by the service.   
  
  
**What is the difference between XMLSerializer and the DataContractSerializer?**   
1. DataContractSerializer is the default serializer fot the WCF   
2. DataContractSerializer is very fast.   
3. DataContractSerializer is basically for very small, simple subset of the XML infoset.   
4. XMLSerializer is used for complex schemas   
  
  
**What is endpoint in WCF service?**   
The endpoint is an Interface which defines how a client will communicate with the service. It consists of three main points: Address,Binding and Contract.   
  
  
**What is the difference between WCF Service and Web Service?**   
1) WCF Service supports both http and tcp protocol while webservice supports only http protocol.   
2) WCF Service is more flexible than web service.   
3) WCF is combination of WebService, Remoting, MSMQ   
  
  
**What are the various ways of hosting a WCF service?**   
Self hosting the service in his own application domain. This we have already covered in the first section. The service comes in to existence when you create the object of ServiceHost class and the service closes when you call the Close of the ServiceHost class. Host in application domain or process provided by IIS Server. Host in Application domain and process provided by WAS (Windows Activation Service) Server.   
  
  
**What is service and client in perspective of data communication?**   
A service is a unit of functionality exposed to the world.   
The client of a service is merely the party consuming the service.   
  
  
**List the behaviors that WCF service uses during its execution.**   
A WCF service uses the following list of behaviors during its execution.   
1. Throttling   
2. Security   
3. Instancing   
- PerCall   
- PerSession   
- Single   
4. Error handling   
5. Concurrency   
- Multiple   
- Single   
- Reentrant   
6. Transactions   
  
  
**What is WCF? Explain.**   
WCF (A platform for SOA)   
Unified programming model, provided by WCF, helps in building Service Oriented Application (SOA) through some simple implementation. WCF Service features for usage in SOA implementation.   
  
Uses of WCF services:   
• Expose functionality using contracts to clients   
• Can be deployed over various protocols to satisfy various distributed and interoperable scenarios   
• Execute autonomously and do not impact another service in case of failure   
• Design and implementation are separate from business logic, which eases migration to SOA design   
  
  
**What is the programming life cycle of WCF?**   
Windows Communication Foundation (WCF) enables applications to communicate whether they are on the same computer, across the Internet, or on different application platforms. This topic outlines the tasks that are required to build a WCF application. The basic tasks to perform, are, in order to:   
1. Define the Service Contract   
2. Implement the Service Contract   
3. Configure the Service by specifying endpoint information and other behavior information   
4. Host the service in an application   
5. Build the client application   
  
  
**Define WCF data contract**   
- A data contract is defined by using a Data Contract Attribute on a class or structure.   
- Members of the data structure which will be used by the service need to be marked with the Data Member Attribute.   
- Only those members will be transferred between the service and its client. In the same way that different classes can implement the same interface, different classes can implement the same Data Contract, and can serialize and deserialize the same data.   
  
  
**What is the advantage and disadvantage of implementing IExtensibleDataObject?**   
WCF guidelines recommend enhancing all data contracts with support of IExtensibleDataObject interface, to preserve unexpected data from clients. During deserialization, superfluous data is placed in a dictionary on the service side and during serialization, the same data is written as XML as it was originally provided by the client. This is very useful to preserve data from version 2.0 services at a version 1.0 client. It is also useful in case where downstream calls from version 2.0 services go to other services handling version 1.0.   
However, there is also a disadvantage of implementing IExtensibleDataObject. It carries risks of denial of service (DoS) and unnecessary use of server resources.   
  
  
**What is the role of WSDL in WCF?**   
WSDL stands for Web Service Description Language. The WCF service exposes the WSDL document for the clients, to generate proxies and the configuration file. The WSDL file provides the following information for the consumers of the WCF service.   
1. Provides the information about the service contract and operations available.   
2. Provides the information about all the end points exposed by the WCF service.   
3. Provides the information about the messages and types that can be exchanged between the client and the WCF service.   
4. WSDL also provides any information about the policies used.   
  
  
**What are the different types of bindings available in WCF?**   
Basic Binding:   
Offered by the BasicHttpBinding class, this is designed to expose a WCF service as a legacy ASMX web service, so that old clients can work with new services. When used by the client, this binding enables new WCF clients to work with old ASMX services.   
  
TCP Binding:   
Offered by the NetTcpBinding class, this uses TCP for cross-machine communication on the intranet. It supports a variety of features, including reliability, transactions, and security, and is optimized for WCF-to-WCF communication. As a result, it requires both the client and the service to use WCF.   
  
Peer Network Binding:   
Offered by the NetPeerTcpBinding class, this uses peer networking as a transport. The peer network-enabled client and services all subscribe to the same grid and broadcast messages to it.   
  
IPC Binding:   
Offered by the NetNamedPipeBinding class, this uses named pipes as a transport for same-machine communication. It is the most secure binding since it cannot accept calls from outside the machine and it supports a variety of features similar to the TCP binding.   
  
Web Service (WS) Binding:   
Offered by the WSHttpBinding class, this usHTTP or HTTPS for transport, and is designed to offer a variety of features such as reliability, transactions, and security over the Internet.   
  
Federated WS Binding:   
Offered by the WSFederationHttpBinding class, this is a specialization of the WS binding, offering support for federated security.   
  
Duplex WS Binding:   
Offered by the WSDualHttpBinding class, this is similar to the WS binding except it also supports bidirectional communication from the service to the client.   
  
MSMQ Binding:   
Offered by the NetMsmqBinding class, this uses MSMQ for transport and is designed to offer support for disconnected queued calls.   
  
MSMQ Integration Binding:   
Offered by the MsmqIntegrationBinding class, this converts WCF messages to and from MSMQ messages, and is designed to interoperate with legacy MSMQ clients.   
  
  
**What are Various Ways of Hosting WCF Services?**   
Three ways of hosting WCF Services:   
1. Self-hosting the service in its own application domain. The service comes into existence when you create the object of Service Host class and the service closes when you call the Close of the Service Host class.   
2. Host in application domain or process provided by IIS Server.   
3. Host in application domain and process provided by WAS (Windows Activation Service) Server.

## Introduction

In this article we will try to understand what are WCF REST services. We will see what is required from a service developers perspective to create a REST enabled WCF service. We see how we can use and consume restful WCF services.

## Background

### Overview of REST

REST stands for Representational State Transfer. This is a protocol for exchanging data over a distributed environment. The main idea behind REST is that we should treat our distributed services as a resource and we should be able to use simple HTTP protocols to perform various operations on that resource.

When we talk about the Database as a resource we usually talk in terms of CRUD operations. i.e. Create, Retrieve, Update and Delete. Now the philosophy of REST is that for a remote resource all these operations should be possible and they should be possible using simple HTTP protocols.

Now the basic CRUD operations are mapped to the HTTP protocols in the following manner:

* **GET**: This maps to the R(Retrieve) part of the CRUD operation. This will be used to retrieve the required data (representation of data) from the remote resource.
* **POST**: This maps to the U(Update) part of the CRUD operation. This protocol will update the current representation of the data on the remote server.
* **PUT**: This maps to the C(Create) part of the CRUD operation. This will create a new entry for the current data that is being sent to the server.
* **DELETE**: This maps to the D(Delete) part of the CRUD operation. This will delete the specified data from the remote server.

so if we take an hypothetical example of a remote resource that contain a database of list of books. The list of books can be retrieved using a URL like:

www.testwebsite.com/books

To retrieve any specific book, lets say we have some ID that we can used to retrieve the book, the possible URL might look like:

www.testwebsite.com/books/1

Since these are GET requests, data can only be retrieved from the server. To perform other operations, if we use the similar URI structure with PUT, POST or DELETE operation, we should be able to create, update and delete the resource form the server. We will see how this can be done in implementation part.

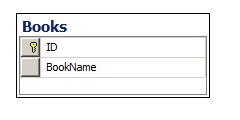
**Note:** A lot more complicated queries can be performed using these URL structures. we will not be discussing the complete set of query operations that can be performed using various URL patterns.

## Using the code

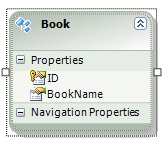
Now we can create a simple WCF service that will implement all the basic CRUD operations on some database. But to make this WCF service REST compatible we need to make some changes in the configuration, service behaviors and contracts. Let us see what WCF service we will be creating and then we will see how we can make useful over the REST protocol.

### creating REST enabled ServiceContract

We will create Books table and will try to perform CRUD operations on this table.



To perform the Database operations within the service lets use Entity framework. This can very well be done by using ADO.NET calls or some other ORM but I chose entity framework. (please refer this to know about entity framework:  [An Introduction to Entity Framework for Absolute Beginners](http://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B)[[^](http://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B)]). The generated Entity will look like following.



Now the service contract will contain functions for CRUD operations. Let us create the ServiceContract for this service:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services)

[ServiceContract]

public interface IBookService

{

[OperationContract]

List<Book> GetBooksList();

[OperationContract]

Book GetBookById(string id);

[OperationContract]

void AddBook(string name);

[OperationContract]

void UpdateBook(string id, string name);

[OperationContract]

void DeleteBook(string id);

}

Right now this is a very simple service contract, to indicate that individual operations can be called using REST protocol, we need to decorate the operations with additional attributes. The operations that are to be called on HTTP GET protocol, we need to decorate them with the WebGet attribute. The operations that will be called by protocols, like POST, PUT, DELETE will be decorated with WebInvoke attribute.

#### Understanding UriTemplate

Now before adding these attributes to these operations let us first understand the concept of UriTemplate. UriTemplate is a property of WebGet and WebInvoke attribute which will help us to map the parameter names coming from the HTTP protocol with the parameter names of ServiceContract. For example, if someone uses the following URI:

localhost/testservice/GetBookById/2

We need to map this first parameter with the id variable of the function. this can be done using the UriTemplate. Also, we can change the function name specifically for the URI and the name of URI function name will be mapped to the actual function name i.e. if we need to call the same URL as:

localhost/testservice/Book/2

then we can do that by specifying the UriTemplate for the operation as:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services)

[OperationContract]

[WebGet(UriTemplate = "Book/{id}")]

Book GetBookById(string id);

Following the same lines, let us define the UriTemplate for other methods too.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services)

[ServiceContract]

public interface IBookService

{

[OperationContract]

[WebGet]

List<Book> GetBooksList();

[OperationContract]

[WebGet(UriTemplate = "Book/{id}")]

Book GetBookById(string id);

[OperationContract]

[WebInvoke(UriTemplate = "AddBook/{name}")]

void AddBook(string name);

[OperationContract]

[WebInvoke(UriTemplate = "UpdateBook/{id}/{name}")]

void UpdateBook(string id, string name);

[OperationContract]

[WebInvoke(UriTemplate = "DeleteBook/{id}")]

void DeleteBook(string id);

}

#### Implementing the Service

Now the service implementation part will use the entity framework generated context and entities to perform all the respective operations.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services)

public class BookService : IBookService

{

public List<Book> GetBooksList()

{

using (SampleDbEntities entities = new SampleDbEntities())

{

return entities.Books.ToList();

}

}

public Book GetBookById(string id)

{

try

{

int bookId = Convert.ToInt32(id);

using (SampleDbEntities entities = new SampleDbEntities())

{

return entities.Books.SingleOrDefault(book => book.ID == bookId);

}

}

catch

{

throw new FaultException("Something went wrong");

}

}

public void AddBook(string name)

{

using (SampleDbEntities entities = new SampleDbEntities())

{

Book book = new Book { BookName = name };

entities.Books.AddObject(book);

entities.SaveChanges();

}

}

public void UpdateBook(string id, string name)

{

try

{

int bookId = Convert.ToInt32(id);

using (SampleDbEntities entities = new SampleDbEntities())

{

Book book = entities.Books.SingleOrDefault(b => b.ID == bookId);

book.BookName = name;

entities.SaveChanges();

}

}

catch

{

throw new FaultException("Something went wrong");

}

}

public void DeleteBook(string id)

{

try

{

int bookId = Convert.ToInt32(id);

using (SampleDbEntities entities = new SampleDbEntities())

{

Book book = entities.Books.SingleOrDefault(b => b.ID == bookId);

entities.Books.DeleteObject(book);

entities.SaveChanges();

}

}

catch

{

throw new FaultException("Something went wrong");

}

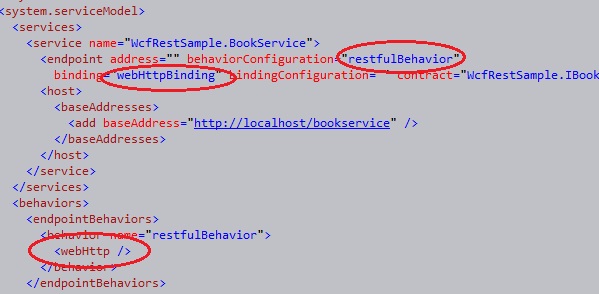
}

}

### Restful WCF service Configuration

Now from the ServiceContract perspective the service is ready to serve the REST request but to access this service over rest we need to do some changes in the service behavior and binding too.

To make the service available over REST protocol the binding that needs to be used is the webHttpBinding. Also, we need to set the endpoint's behavior configuration and define the webHttp parameter in the endpointBehavior. So our resulting configuration will look something like:

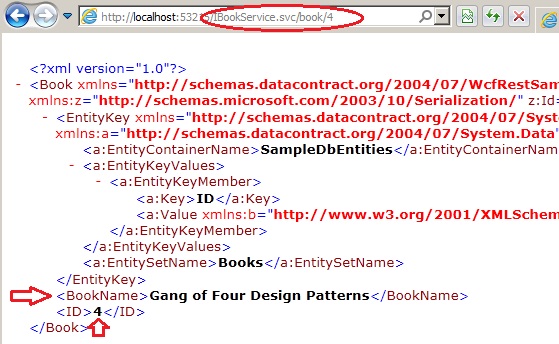


#### Test the service

Now to test the service we will simply run the service and use the URLs to retrieve the data. let see this for our GET operations in action.



And now testing the query to get a single record



And so we have seen that we received the response in the browser itself in form of XML. We can use this service without even consuming it by adding a service reference by using the URLs and HTTP protocols.

**Note:** Here I am not demonstrating the other operations for POST, PUT and DELETE but they are fairly straight forwards and a simple HTML page sending the data using the required protocol with the specified parameter names will perform the operation.

### Using JSON

We can also change the Response and Request format to use JSON instead of XML. To do this we need to specify properties of the WebInvoke attribute.

* RequestFormat: By default its value is WebMessageFormat.XML. to change it to JSON format, it needs to be set to WebMessageFormat.Json.
* ResponseFormat: By default its value is WebMessageFormat.XML. to change it to JSON format, it needs to be set to WebMessageFormat.Json.

Let us create one more operation in our service contract called as GetBooksNames and will apply the ResponseFormat as Json for this method.

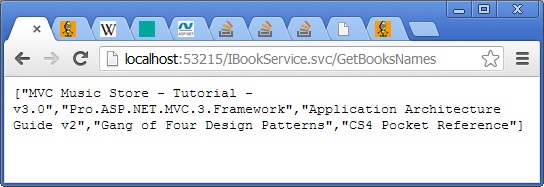
http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services)

[OperationContract]

[WebGet(ResponseFormat=WebMessageFormat.Json]

List<string> GetBooksNames();

The response will now appear in the JSON format.



And now we have a WCF REST service ready with us.

**Note:** We also have a ready made template in Visual studio to create WCF data services that provides us with easy way to create REST enabled ODATA services. We will perhaps talk about them separately.

## Introduction

Windows Communication Foundation (WCF) is an SDK for developing and deploying services on Windows. WCF provides a runtime environment for your services, enabling you to expose CLR types as services, and to consume other services as CLR types. In this article, I am going to explain how to implement restful service API using WCF 4.0 . The Created API returns XML and JSON data using WCF attributes.

## What is REST?

Based on the Roy Fielding theory "Representational State Transfer (REST), attempts to codify the architectural style and design constraints that make the Web what it is. REST emphasizes things like separation of concerns and layers, statelessness, and caching, which are common in many distributed architectures because of the benefits they provide. These benefits include interoperability, independent evolution, interception, improved scalability, efficiency, and overall performance."

Actually only the difference is how clients access our service. Normally, a WCF service will use SOAP, but if you build a REST service, clients will be accessing your service with a different architectural style (calls, serialization like JSON, etc.).

REST uses some common HTTP methods to insert/delete/update/retrieve information which is below:

1. **GET** - Requests a specific representation of a resource
2. **PUT** - Creates or updates a resource with the supplied representation
3. **DELETE** - Deletes the specified resource
4. **POST** - Submits data to be processed by the identified resource

## Why and Where to Use REST?

Few days back, I was writing a service which was supposed to be accessed by heterogeneous language/platform/system. It can be used by iPhone, Android, Windows Mobile, .NET web application, JAVA or PHP. Using web service, it was bit complex for me to expose it to everyone using uniform system. Then we decided to use REST, which was easily espoused over cloud. This was a great example which shows the capability of **SIMPLE RESTful SERVICE** :). Below are some points which will help you to understand why to use the RESTful services.

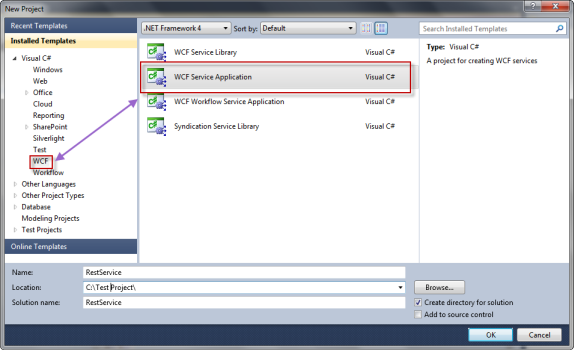
1. Less overhead (no SOAP envelope to wrap every call in)
2. Less duplication (HTTP already represents operations like DELETE, PUT, GET, etc. that have to otherwise be represented in a SOAP envelope).
3. More standardized - HTTP operations are well understood and operate consistently. Some SOAP implementations can get finicky.
4. More human readable and testable (harder to test SOAP with just a browser).
5. Don't need to use XML (well, you kind of don't have to for SOAP either but it hardly makes sense since you're already doing parsing of the envelope).
6. Libraries have made SOAP (kind of) easy. But you are abstracting away a lot of redundancy underneath as I have noted. Yes, in theory, SOAP can go over other transports so as to avoid riding atop a layer doing similar things, but in reality just about all SOAP work you'll ever do is over HTTP.

## Step by Step Guide

Generally, a developer is scared to use WCF because of a lot of confusing configuration. I will try to use minimum configuration so that it will be easier to understand for us. We will develop **Restful WCS API** in 6 steps. So let’s start now.

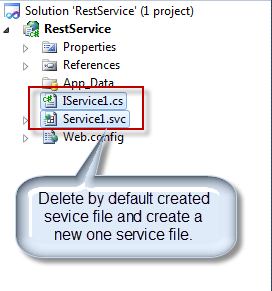
#### STEP 1

First of all launch Visual Studio 2010. Click **FILE**->**NEW**->**PROJECT**. Create new "**WCF Service Application**".



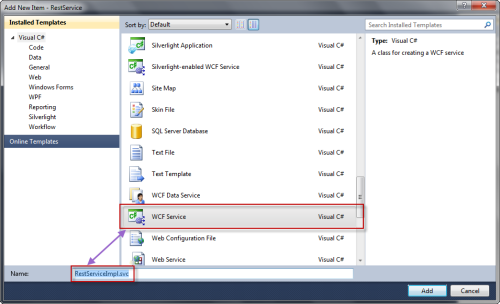
#### STEP 2

Once you create the project, you can see in solution that By Default WCF service and interface file are already created. Delete By default created file as we will create our own interface and WCF service file.



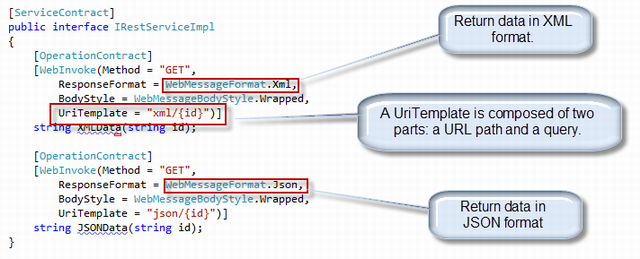
#### STEP 3

Now right click on solution and create one new WCF service file. I have given name to the service file as “***RestServiceImpl.svc***”.



#### STEP 4

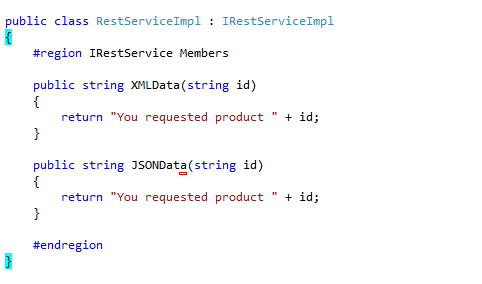
As I explained at the start of the article that we will be writing an API which can return data in XML and JSON format, here is the interface for that. In IRestServiceImpl, add the following code:



In the above code, you can see two different methods of IRestService which are XMLData and JSONData. XMLData returns result in XML whereas JSONData in JSON.

#### STEP 5

Open the file RestServiceImpl.svc.cs and write the following code over there:



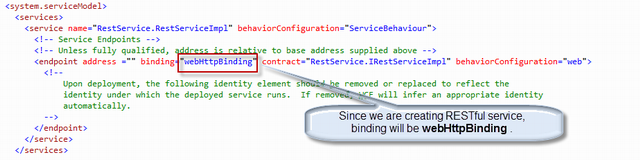
#### STEP 6

Now let’s move to configuration part which is the last one. There will be two basic parts of the configurations file which we must have to understand.

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/105273/Create-RESTful-WCF-Service-API-Step-By-Step-Guide)

<services>

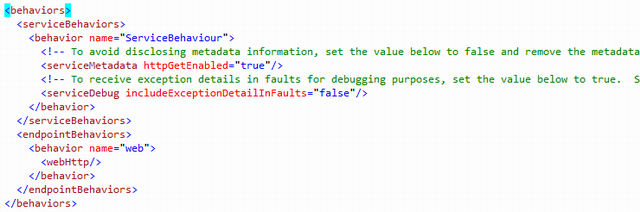
This part contains information about the End Point. Below are the code details.

[](http://www.codeproject.com/KB/WCF/RestServiceAPI/image6.png)

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/105273/Create-RESTful-WCF-Service-API-Step-By-Step-Guide)

<behaviors>

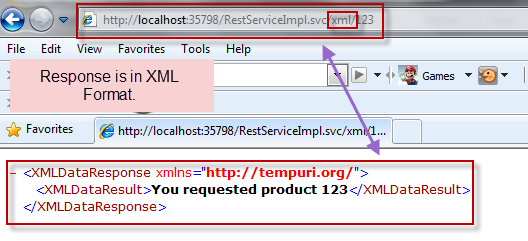
This part contains details about service and endpoint behavior.



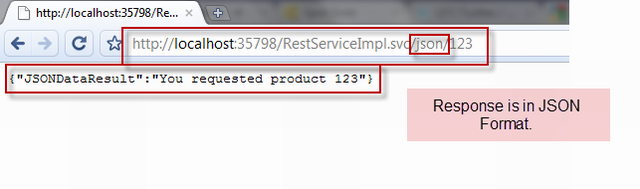
And that’s it. Our Restful WCF service is ready for test purposes.

## Service Ready to Test Now

Now I launch the application in the browser to see the result. I launch this service in Internet Explorer and my URL is now <http://localhost:35798/RestServiceImpl.svc>. Now if I use <http://localhost:35798/RestServiceImpl.svc/xml/123> URL, I get the following response on the browser which is an XML format and that was my task to achieve.



Now if I use <http://localhost:35798/RestServiceImpl.svc/json/123> URL, I get the following response on the browser which is an XML format and that was my task to achieve.



In this article we will discuss Message Contracts. We will also discuss MessageContractAttriubtes, MessageHeaderAttriubtes, MessageBodyMemberAttriubtes and a sample code demonstration.  
  
In WCF services, a DataContract enables us to define the structure of the data. This data is sent in the body of our SOAP (Simple Object Access Protocol) messages. These messages may be of either inbound (request) or outbound (response) type. A message is nothing but a packet and WCF uses this packet to transfer information from source to destination. This message contains an envelope, header and body.  
  
  
  
A Message Contract is used to control the structure of a message body and serialization process. It is also used to send / access information in SOAP headers.   
  
**Message Contract Attributes**

* MessageContractAttribute
* MessageHeaderAttribute
* MessageBodyMemberAttribute

**MessageContractAttribute**  
  
The MessageContract Attribute is applied to a class or structure to define our own message structure, as in:

[MessageContract()]

public class AuthorRequest

{

}  
  
**Properties of MessageContractAttribute**

1. public bool HasProtectionLevel { get; }  
     
   HasProtectionLevel gets a value that indicates whether the message has a protection level.  
     
   It returns true if the message must be encrypted, signed, or both; otherwise false. The default is false.
2. public bool IsWrapped { get; set; }  
     
   IsWrapped gets or sets a value that specifies whether the message body has a wrapper element.  
     
   It returns true if the message body has a wrapper element; otherwise, false. The default is true.
3. public ProtectionLevel ProtectionLevel { get; set; }  
     
   ProtectionLevel gets or sets a value that specified whether the message must be encrypted, signed, or both.  
     
   It returns one of the System.Net.Security.ProtectionLevel values. The default is   
   System.Net.Security.ProtectionLevel.None.
4. public string WrapperName { get; set; }  
     
   WrapperName gets or sets the name of the wrapper element of the message body.  
     
   It returns the name of the wrapper element in the message body.
5. public string WrapperNamespace { get; set; }  
     
   WrapperNamespace gets or sets the namespace of the message body wrapper element.  
     
   It returns the wrapper element namespace.

For example:

[MessageContract(IsWrapped = false,ProtectionLevel=ProtectionLevel.None)]

public class AuthorRequest

{

}

**MessageHeaderAttribute**  
MessageHeaderAttribute is applied to the members of the MessageContract to declare the members within the message header; see:

[MessageContract(IsWrapped = false,ProtectionLevel=ProtectionLevel.None)]

public class AuthorRequest

{

    [MessageHeader()]

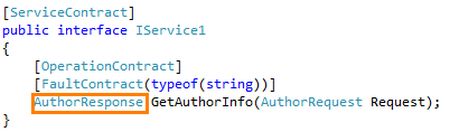
    public string AuthorId;

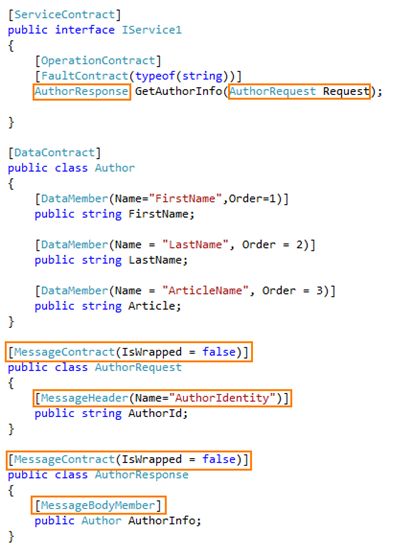
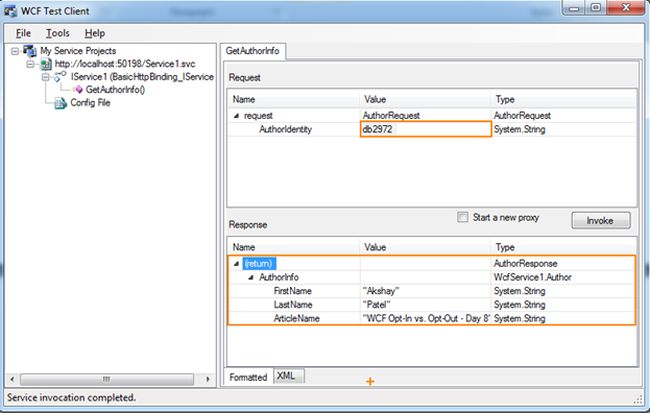
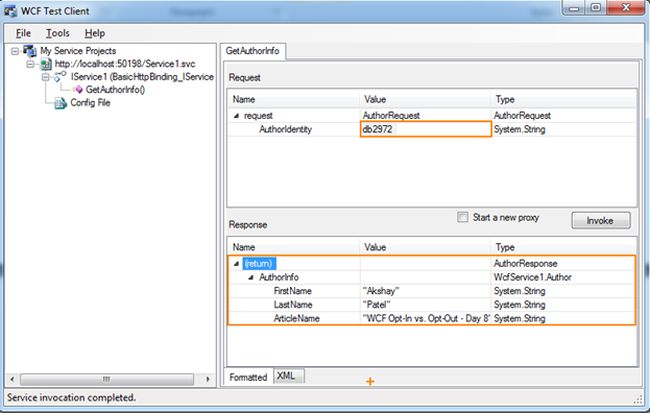
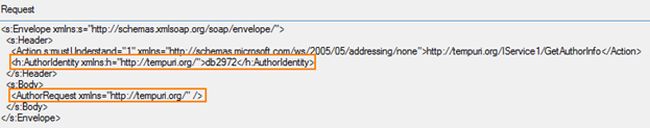
}

**Properties of MessageHeaderAttribute**

1. public string Name { get; set; }  
     
   Name specifies the name of the element that corresponds to this member.  
     
   It returns the name of the element that corresponds to this member. This string must be a valid XML element name.
2. public string Namespace { get; set; }  
     
   Namespace specifies the namespace of the element that corresponds to this member.  
     
   It returns a namespace URI of the element that corresponds to this member.
3. public ProtectionLevel ProtectionLevel { get; set; }  
     
   ProtectionLevel specifies whether the member is to be transmitted as-is, signed, or signed and encrypted.  
     
   It returns one of the System.Net.Security.ProtectionLevel values. The default is System.Net.Security.ProtectionLevel.None.
4. public string Actor { get; set; }  
     
   Actor gets or sets a URI that indicates the node at which this header is targeted. It maps to the role header attribute when SOAP 1.2 is used and the actor header attribute when SOAP 1.1 is used.  
     
   It returns a URI that indicates the node at which this header is targeted. This URI maps to the role header attribute when SOAP 1.2 is used and the actor header attribute when SOAP 1.1 is used.
5. public bool MustUnderstand { get; set; }  
     
   MustUnderstand specifies whether the node acting in the System.ServiceModel.MessageHeaderAttribute.Actor role must understand this header. This is mapped to the mustUnderstand SOAP header attribute.  
     
   It returns true if the node acting in the System.ServiceModel.MessageHeaderAttribute.Actor role must understand this header; otherwise, false.
6. public bool Relay { get; set; }  
     
   Relay specifies whether this header is to be relayed to downstream nodes. This is mapped to the relay SOAP header attribute.  
     
   It returns true if this header is to be relayed to downstream nodes; otherwise, false.

**MessageBodyMemberAttribute**  
  
MessageBodyMemberAttribute is applied to the members of message contracts to declare the members within the message body.  
  
**Properties of MessageBodyMemberAttribute**  
  
Name, Namespace, ProtectionLevel and Order are the properties of MessageBodyMemberAttribute. As we are now familiar with all the properties, I am not explaining here once again.  
  
**Why should we use MessageContract**  
  
Suppose we are not using MessageContract in our service, by default SOAP body element contains a child wrapper element of operation name and wraps parameters. If there is no parameter then this element will be empty in the inbound (request) message. In the same way, in an outbound (response) message, if it is of type void or say nothing to return then this element will be empty.  
  
For example we create a service and set MessageContract with the property IsWrapped to false. Then there is no child element in the SOAP body. You will see a collection of MessageBodyMembers directly under the SOAP body within the MessageContract.   
  
Controlling wrapping is important when we deal with the other platforms except WCF because they might serialize their SOAP messages differently.   
  
**Sample Code**  
  
Consider a scenario where we create a service and in this service we have one operation contract which returns the author information. Now we want to set some authentication like AuthorId should be matched. In this case we store AuthorId in MessageHeader because it is safe. Here we create two MessageContracts, one is for the request i.e. AuthorRequest and another is for the response i.e. AuthorResponse.  
  
While creating a MessageContract consider the following two points:

1. When we pass a parameter of type MessageContract, only one parameter is used in the service operation.  
     
   
2. The return type of the OperationContract is of MessageContractType or Void type.  
     
   

Now add the following lines of code to your Interface.  
  
  
  
Now add an implementation of the above operation contract in the service class.  
  
  
  
**Test Client Output (IsWrapped = false)**  
  
Insert 'db2972' as the input for the AuthorIdentity parameter; see:  
  
  
  
**SOAP Request**  
  
An AuthorIdentity element is in the SOAP request as we pass this under the MessageHeader. And the Body element is empty because we are not passing any value.  
  
  
  
**SOAP Response**  
In response we get author information directly under the body element as we set IsWrapped to false.  
  
  
  
**Test Client Output (IsWrapped = true)**  
Now change the code, set IsWrapped to true in the MessageContract and once again check the result in Test Client.  
  
  
  
**SOAP Request**  
AuthorRequest element is empty and is added under the body element because we set IsWrapped to true.  
  
  
  
**SOAP Response**  
In the same way in the response also the AuthorResponse element is added under the body element.  
  
